JPRS-CST-93-004
3 March 1993

FOREIGN BROADCAST INFORMATION SERVICE

JPRS Report

Science & Technology

China

Artificial Neural Networks

REPRODUCED BY
U.S. DEPARTMENT OF COMMERCE
NATIONAL TECHNICAL INFORMATION SERVICE
SPRINGFIELD, VA. 22161

19980115 028

Science & Technology

China

Artificial Neural Networks

JPRS-CST-93-004 *3 March 1993*

CONTENTS

General-Purpose Master-Slave Neural Network Model

93FE0193A Beijing DIANZI XUEBAO [ACTA ELECTRONICA SINICA] in Chinese
Vol 20 No 10, Oct 92 pp 26-32, 60

[Article by Chen Guoliang [7115 0948 5328], Song Songchun [1345 2646 4783], and Qin Xiaoou [4440 1420 7743] of the Department of Computer Science, University of Science and Technology of China (USTC): "General-Purpose Master-Slave Neural Network Model"; MS received Apr 92, revised Jul 92, supported by the National 863 Program]

[Excerpts]

Abstract

A new neural network model, a general-purpose master-slave neural network model, is presented. The general-purpose nature of this model is proven by two master-slave control methods.

Key words: master-slave neural network, energy function, local minimum, system stability.

I. Introduction

Just as with earlier automatons, existing artificial neural networks are dedicated systems (counting special symbols for example). General-purpose computers began to emerge after the concept of "stored program" was introduced. The program determines the sequence of information processing. If this concept is applied to a neural network, we will find that its function is determined by the architecture of the network (which determines the processing sequence of network information) and degree of excitation of neuron. To this end, the key step in the design of a general-purpose neural network is to build a master control module. Its output can be used to regulate the weighted value of the slave module, or to "clamp" the degree of excitation of the slave module. The structure of the main module is "programmed" to define its function. Figure 1 shows a simple master-slave neural network we have designed. The master module controls the weighted value of the slave module by way of several intermediate channels. The topological architecture of the slave module is arbitrary. (The slave network shown in the figure has a layered structure.)

The general-purpose master-slave neural network proposed can be controlled in two different methods. (1) The output of the master network controls the degree of excitation of slave neural network elements in the form of an external current. (2) The master network output regulates the link right of the slave network. In reality, (1) and (2) may be combined in use.

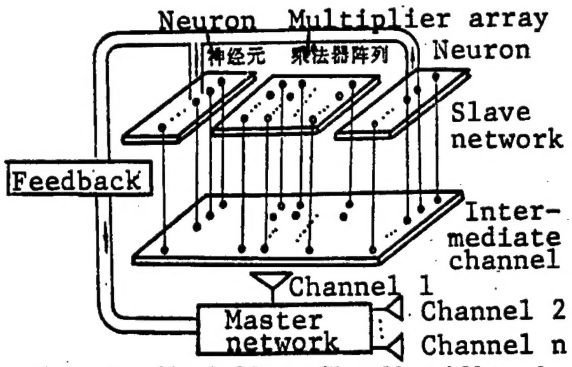


Figure 1. A Simple Master-Slave Neural Network

Specifically with reference to these two basic methods, two master-slave models are designed and simulated on a computer.

II. General-Purpose Master-Slave Model With Link Weight Regulation

A BP network is essentially a nonlinear mapping function:[1]

$$Y = F_1(W_1 \cdot F_2(W_2 \cdots\cdots X)) \qquad (1)$$

where X and Y are the input and output vector of the network, and W_j is the link weight matrix between the jth and j + 1th layer.

The shortcoming of the present BP [backpropagation] network is its slow learning process. It is difficult to use hardware to reflect the change in the weight. Furthermore, the learning algorithm essentially is a nonlinear optimization process. There is not a good method for overcoming difficulties associated with local minima. Specifically in response to these problems, the network shown in Figure 2 is introduced. A Hopfield network with m x n neurons is used to replace the weight matrix $W_{m \times n}$ for every layer of the BP network. The output of each neuron V_{ij} represents weight W_{ij} of the original BP network. Since the nature of a Hopfield network is that the energy function of the network is minimized when it converges,[2] it is possible to utilize this behavior of the Hopfield network to help converge the BP network if the error function of the BP network is treated as the energy function of the Hopfield network. [passage omitted]

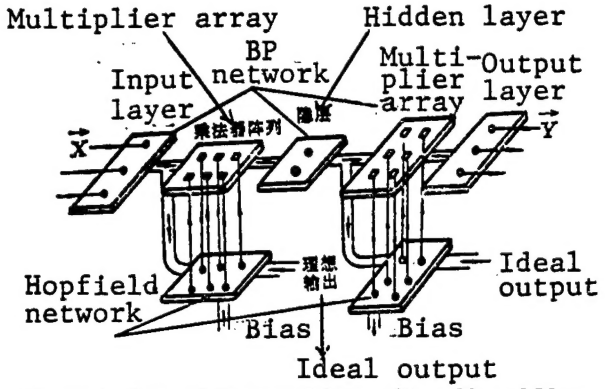


Figure 2. General-Purpose Master-Slave Neural Network With Tunable Weighted Interconnection

A control layer is added on top of the Hopfield layer. A bias current is injected to cause the error function of the BP network to jump at the local minimum. This technique is used in the other model discussed in this paper.

By now, the following conclusions can be reached for this model:

(1) From equation (3), regardless of whether it is a BP network or Hopfield network, the weight is expressed in terms of the output state of the neuron. Thus, the model is not relevant to the actual problem and the network is more general-purpose in nature. It is relatively easy to realize a variety of nonlinear mappings.

(2) The model can easily be constructed in hardware because weight link matrices can be realized by means of multipliers. Compared to using resistors or RAM, it is more flexible and accurate.

(3) The learning process of the BP network is automatically completed under the control of the Hopfield layer.

This model is used to solve the "mirror image symmetry" recognition problem presented by D. E. Rumelhart.[3] When an n-dimensional vector comprised of n symbols is applied to the input end of the network, the output should point out whether this string is symmetric left to right (for example, 00111100 is a symmetric string). This is a classic problem for testing the BP algorithm. It imposes a high degree of stability on the BP network. The data provided by Rumelhart show that the network needed to be trained approximately 90,000 times before converging when n = 6. In our model, the nonlinear mapping function of the Hopfield network is chosen to be $f(u) = 1/(1 + \exp(-\alpha u))$. When $0.2 < \alpha < 2$, $0.1 < \varepsilon < 0.8$ (where ε is the iteration step given in equation (4)), the network converges after 32,000-60,000 times of training. Figure 3 shows α and ε as a function of number of times to reach convergence. Apparently, this number is related to the product of α and ε; when the product of α and ε is a specific value, the number required to converge is minimized. The farther it is off from this value, the larger the number of times required to converge; or, it may not converge at all.

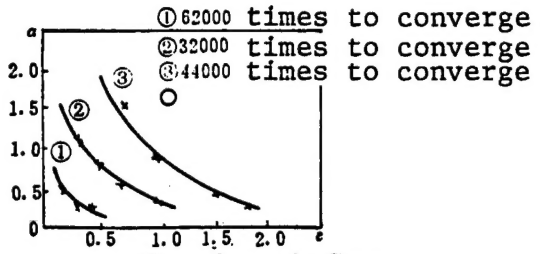


Figure 3. α and ε Curve

III. General-Purpose Master-Slave Model Using External Current Control

In order to solve the NP problem in conventional optimization, i.e., the traveling salesman problem (TSP), a master-slave neural network such as the one shown in Figure 4 has been designed. The slave network is a Hopfield network and the master network is a conventional computer. Based on the link matrix, an effective Lyapunov function has been designed. The convergence of the new model and the validity of the solution are verified by way of analysis of the Eigen value of the network link matrix and the network dynamic equation. [passage omitted]

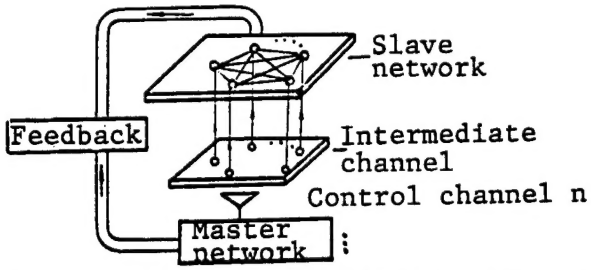


Figure 4. Master-Slave Model Using External Current Control

IV. Conclusions

The master-slave models presented in this paper not only are capable of delineating the characteristics of the complicated biologic neural system (master-slave differentiation)[8] but also are to some extent general-purpose in nature by stressing the use of a conventional computer. Master-slave differentiation indicates that information is processed separately based on its relevance. This will reduce the communications load and makes it feasible to build a coarse artificial neural network processor.

References

1. Rumelhart, D. E., McClelland, J. L., "Parallel Distributed Processing," Vol 1-2, MIT Press, 1986.

2. Hopfield, J. J., Tank, D. W., "Neural Computation of Decision in Optimization Problems," BIO. CYBERN., 1985, 52: 141-152.

3. Rumelhart, D. E., Hinton, G. E., Williams, R. J., "Learning Representations by Back-Propagation Errors," NATURE, 1986, 323 (9): 533-536.

4. Chen Guoliang, "Using Neural Network for Optimization of Combinations," C^2N^2, Proceedings of First Neural Network Conference in China, 1990, Beijing, pp 122-129.

5. Song Songchun, Tang Xinan [0781 6932 0589], and Chen Guoliang, "An Adaptive Hopfield Neural Network and Its Dynamic Analysis," National Artificial Intelligence and Intelligent Computer Conference, Beijing, 1991, pp 136-143.

6. Cardon, H., "A NN for Solving TSP on the Basis of City Adjacency in the Tour," CUED, 1990.

7. Tang, Xi-nan, et al., "Equivalent-Class Energy Function for Solving the TSP," Proc. Int'l Conf. for Young Computer Scientists, Beijing, 1991.

8. Norman Geschwind, "The Specialization of Brain," SCIENTIFIC AMERICAN, 1980, 1.

Chen Guoliang, Professor and Chairman of Department of Computer Science at USTC, is engaged primarily in areas such as parallel/distributed processing and algorithms, intelligent computer architectures, neural networks and neural computation theory.

Song Songchun, a student in the Dept. of Computer Science at USTC, is conducting research in neural computation theory under the direction of Professor Chen Guoliang.

Qin Xiaoou, a student in the Dept. of Computer Science at USTC, is conducting research in neural network applications under the direction of Professor Chen Guoliang.

Design, Applications of Programmable Neuro-Chip With Floating-Gate NMOS Transistors

93FE0193B Beijing DIANZI XUEBAO [ACTA ELECTRONICA SINICA] in Chinese Vol 20 No 10, Oct 92 pp 50-55

[Article by Wang Yang [3769 7122] of the Institute of Microelectronics, Beijing University, and Li Zhijian [2621 1807 1017] and Shi Bingxue [4258 4426 1331] of the Institute of Microelectronics, Qinghua University: "Design, Applications of Programmable Neuro-Chip With Floating Gate NMOS Transistors"; MS received Mar 92, revised Jun 92, supported by the National Natural Science Foundation]

[Text]

Abstract

A floating-gate NMOS transistor neural network has been designed. Its features include simple structure, small chip area and continuously adjustable interconnection weight. In addition, it possesses the characteristics of distributed neurons and can be cascaded to form a large network. An 8 x 8 interconnected neuro-chip has been fabricated using a 3 μ m floating-gate NMOS process. The chip contains 128 programmable floating-gate NMOS transistors, which is equivalent to a fully interconnected network comprised of 8 neurons. It has been used in number recognition and binary image processing. The results show that the network has a great deal of potential. Furthermore, it is highly flexible and easy to fabricate because of its simple structure and adaptability to IC technology.

Key words: artificial neural network, IC, circuits principle and design.

I. Introduction

Theoretical analysis of neural networks is still being conducted. Research on neural networks must still rely on analog tools. Hardware support is required in solving complex problems involving large-scale parallel systems. Hence, a lot of interest is focused on the implementation of neural networks. A large number of electronic methods are available to implement neural networks. In the area of VLSI, there are digital circuits,[1] analog circuits,[2] digital/analog circuits,[3], voltage circuits, current circuits,[4] pulse current circuits,[3] etc. The major feature of a neural network is the formation of a large nonlinear dynamic system through the interconnection of a large number of simple elements. When a network is implemented for a parallel system, the element ought to be simple in structure and the chip should be small in order to form a large-scale network. The high computing power and flexibility of a neural network comes mainly by way of adjusting the synaptic weights. It is imperative to be able to fabricate neural networks that have continuously adjustable weights.

The basic element of a neural network not only is an operator but also a storage device. Its complexity and area are determined by both operation and storage requirements. As far as operation is concerned, most conventional methods are based on theoretical algorithms and do not fully utilize the physical characteristics of the device. It is very difficult to arrive at a simple design. Active devices in a digital circuit behave as switches. A device operates in the saturation region and cutoff region only and its characteristics are poorly utilized. When the computation is complete, the circuit is complicated, the chip area is large and the computation time is long. Nevertheless, this kind of circuit has a high level of tolerance to noise and is the most mature circuit in use. In an analog circuit, an active device operates in a linear region. The physical laws in the linear region may be utilized in performing the computations. Compared to digital circuitry, its utilization rate

is higher, the chip area is smaller and the computing speed is faster. However, the circuit is more susceptible to noise. Even though the accuracy of each individual device is sufficiently high, due to the accumulation of errors the final result is severely impacted by noise when the entire computation is completed. Therefore, it is difficult to make a large-scale computation system with this kind of circuitry. Since the correctness of a neural network not only depends on the accuracy of each individual device but also on the feedback of the network, it does not impose absolutely precise device parameters, rigorously matched devices and accurate time constants. We should be able to take more advantage of the physical characteristics of the device compared to an analog circuit (i.e., letting the transistor operate over its entire characteristic region) so that the basic computation element has the simplest structure and occupies the least amount of area. The storage of analog data at the basic element level is one of the key issues in the implementation of neural networks. Its behavior also determines the performance of the entire network. The floating-gate structure is a simple and low-chip-area analog storage device. It consumes little power for erasing and writing. The charge leakage of floating-gate storage is also low, which is especially suitable for the storage of synaptic weights.

A simple and highly flexible neuro-chip has been designed and fabricated based on 3 μ m floating-gate NMOS process. It employs the floating-gate NMOS device to perform the computation and storage function of the basic element. This not only greatly simplifies the programmable network but also makes it adaptable to IC technology. A model was built for number recognition and it is used as an example to demonstrate its application as a Hamming network. It was also used in noise cancellation and edge detection of binary images to illustrate the fact that the chip might be used in variable threshold circuits.

II. Design and Implementation of Neuro-Chip

Figure 1 shows the neuron of a network comprised of floating-gate NMOSFETs. Since all neurons are identical in structure in this network, they essentially have the same conductivity factor and parasitic capacitance. Hence, the interconnect strength can only be adjusted by threshold voltage. The dynamic equation describing this kind of a nonlinear network is as follows:

$$\frac{dU_i}{dt} = K\left[\sum_{j=1}^{N} F_{ij}(U_j, U_i, V_{T_{ij}}) + \sum_{j=N+1}^{N+M} F_{ij}(X_{j-N}, U_i, V_{T_{ij}})\right]$$

$$F_{ij}(U_j, U_i, V_{T_{ij}}) = \begin{cases} f(U_j - U_i, V_{DD} - U_i, V_{Tij}) & \text{excited interconnect} \\ -f(U_j, U_i, V_{Tij}) & \text{inhibited interconnect} \end{cases}$$

$$i = 1, \cdots, N; \quad j = 1, \cdots, N+M$$

where K is the ratio of transistor conductivity factor to neuron input capacitance, V_{DD} is the power supply voltage, N is the number of neurons, M is the external input number into the network, and f is the characteristic function of the floating-gate transistor. The first term inside the bracket on the right hand side of the above equation represents the network feedback and the second term represents the feed-forward part of the network. This neural network has been constructed based on the nonlinear characteristics of the transistor.

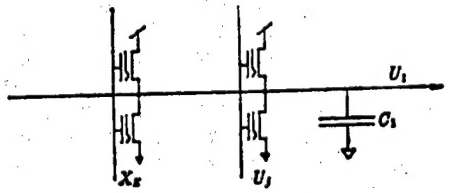


Figure 1. Floating-Gate NMOS Neuron Structure

Figure 2 shows the circuit of the floating-gate NMOS transistor neural network based on such a neuron structure. It is comprised of three parts. The first part is the basic network which contains 8 x 8 basic interconnected elements. The second part controls the operation that includes $M_{c1\text{-}8}$, $M_{g1\text{-}8}$ and $M_{i1\text{-}8}$. $M_{c1\text{-}8}$ control the interconnect states. When φ_C is at a high voltage, the network operates in a feedback mode and when φ_C is at a low voltage, the network operates in a non-feedback mode. $M_{g1\text{-}8}$ clear the neuron dendrite lines (output). $M_{i1\text{-}8}$ control the signal input. In addition, their presence avoids the direct connection of the grid of the neuron to the input leg to protect the programming in the network when φ_i is low. The third part is a program control circuit that is comprised of $M_{o1\text{-}8}$ and $M_{e1\text{-}8}$. They switch to control the program to choose between excited or inhibited interconnect. When φ_o is high and φ_e is low, the program is excitedly interconnected. When φ_o is low and φ_e is high, the interconnect is inhibited.

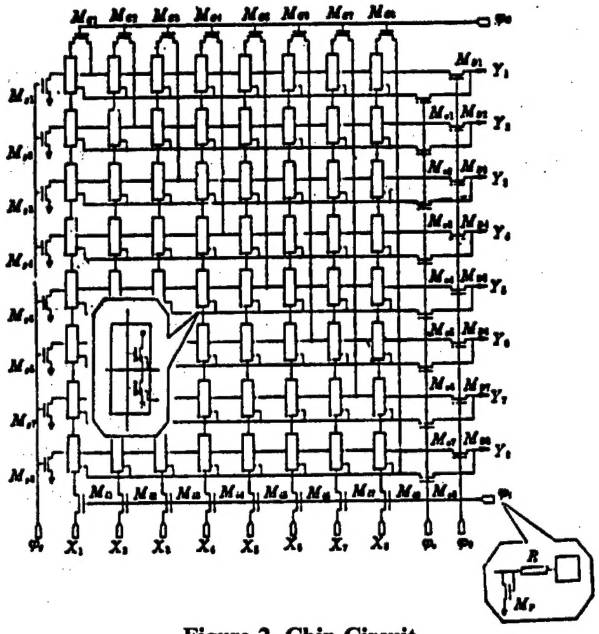


Figure 2. Chip Circuit

This NMOS transistor neural network has the feature of a distributed neuron structure. One disadvantage of the conventional resistor amplifier neural network is that the fan-in and fan-out of the amplifier must be increased as the network expands. Furthermore, its ability to amplify the lowest signal cannot be reduced. This makes the design of the amplifier difficult. Hence, a distributed neuron network structure[6] was proposed. The number of amplifiers is increased in order to reduce the need to enhance the fan-in and fan-out capability of each amplifier. From a certain angle, each transistor in a transistor neural network accomplishes the function of a resistor and an amplifier in a distributed neuron network. Since this network possesses the features of a distributed neuron network, it may be conveniently cascaded into a larger network, or a network of a different structure. Thus, the adaptability and flexibility of the hardware are improved. In addition, this also facilitates the design of the neural network on the circuit board.

In order to shorten the leads, the output lines are designed to be horizontal and in the chip layout. The input lines are perpendicular to the output lines. Each basic element in the network consists of two floating-gate NMOSFETs. One is the excited interconnect between the programming electrode and the source and the other is the inhibited interconnect that has an independently programmed electrode. In order to have a symmetric excited and inhibited interconnect, to the extent possible, both transistors have the same master pattern.

The circuit on the 3 μ m floating-gate NMOS chip is 2.5 x 3.3 mm^2 in area. It includes 128 programmable NMOSFETs and over 50 peripheral transistors, as shown in Figure 3 [photograph not reproduced]. Results measured show that the breakdown voltage of transistors in the peripheral circuit ensure that the programmed voltage can be applied to the basic element. The programming characteristics of the basic interconnect element is described in reference 7.

III. Hamming Network Constructed With Neural Network Chips

The Hamming network is a double-layer network comprised of a maximum network and a template-matching network. It can compare an input vector with existing template vectors and determine the matching template with the minimum Hamming distance. This is a common operation in pattern recognition.

A beneficial result of biologic neural network research is the discovery of a side inhibiting effect. One of the basic modes of the neural network established based on this principle is a competitive network. The survival-of-the-fittest network[8] is a classic competitive network. Its function is to locate the maximum among a set of input data by way of network operation. The neuron corresponding to the maximum input has the maximum output by means of competitive evolution while other output values become the minimum. When implementing this function with a chip circuit, the network is operating in a full-feedback mode. The interconnects between a certain neuron and other neurons are inhibited, while the connection to itself is excited. In a transistor neural network, an interconnecting transistor not only adjusts the weight but also amplifies. The relative control of the output current by each transistor reflects the strength of the interconnect matrix. The absolute control of the output current is a function of the electrical properties (such as high-voltage or low-voltage output, time delay, etc.) of the whole network.

Another important application of the neural network is to compare an input vector to template vectors already stored in the network to calculate the degree of matching between the input vector and the template vector. In principle, this computation can be accomplished by three different interconnect methods.[5] As for the survival-of-the-fittest maximum network constructed by the chips described earlier, it is more appropriate to employ inhibitively interconnected elements to store the template-matching network.

By connecting a survival-of-the-fittest network to a matching network, we have a Hamming network. The matching template for numbers 1-8 is as shown in Figure 4. Since the template is a 15-dimensional vector, it requires two 8 x 8 network chips to complete the computation for template similarity. Furthermore, an 8 x 8 network chip is required to find the maximum for eight matching templates. Therefore, the Hamming network for number recognition requires three chips. Figure 5 shows the circuit for the entire system. By means of weighted interconnection through programming, the three chips are weight-adjusted to be connected in the

manner shown in the circuit. The figure does not show any interconnects that correspond to the case where the threshold voltage is higher than the source voltage because they are not functional in the circuit. The weight-adjusted circuit was linked to a computer to conduct number recognition tests. Figure 6 shows the results of a series of experiments. For every pair of numbers in the figure, the left one represents the input vector and the right one is the final recognition result.

Figure 4. Matching Templates for Numbers 1-8

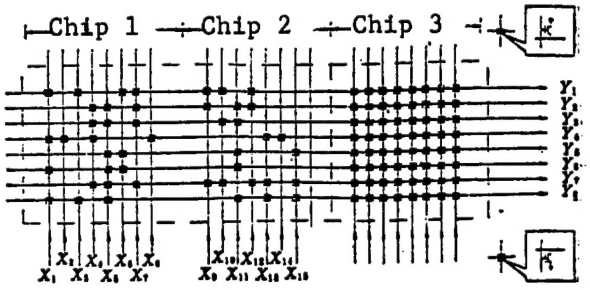


Figure 5. Circuit of Hamming Network for Recognition of Numbers 1-8

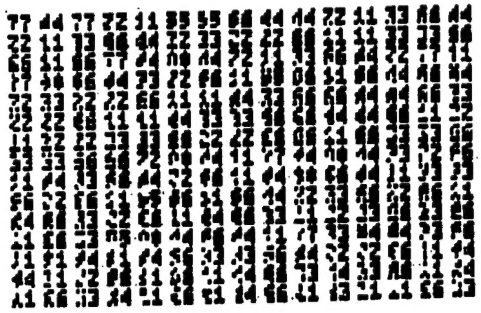


Figure 6. Experimental Result of Number Recognition

IV. Variable-Threshold Logic Network Constructed With Chip Circuit

The threshold logic concept was presented by McCulloch and Pitts[9] to describe neuron mathematical models and Boolean functions. Variable-threshold logic is threshold logic with variable interconnect weight and a variable threshold. The threshold element is a logic element with a binary input and an output value of either "0" or "1." Its property is determined by the weight and the threshold. A single-threshold element can only be used to implement a linear separable function. A network comprised of threshold elements can be used to implement any arbitrary logic function.

The transistor neuro-chip can conveniently be used as a variable-threshold element. Its circuit is shown in Figure 7. The threshold voltage of inhibitively interconnected transistors M_{I1}-M_{IN} determines the weight and the threshold voltage of transistor M_T determines the threshold of the element. M_{c11}-M_{c22} perform operations such as weighted input and threshold comparison. The output of this threshold element is complementary.

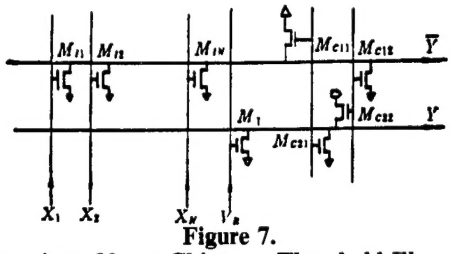


Figure 7.
Transistor Neuro-Chip as a Threshold Element

As an example, a variable-threshold logic network has been used to eliminate noise from a binary image. A 3 x 3 pixel window is moved sequentially across a binary image. The number of black pixels in the window at each position is counted. Corresponding to the new image, the output is black only when the number of counts of the window is greater than or equal to the threshold. For such a noise-cancellation method, it is simpler to accomplish with a logic circuit. Because the logic function in this method is linear and separable, it can be easily implemented with a threshold logic circuit.

Figure 8 shows the results of the actual treatment. Figure (a) shows the noisy input binary image; (b), (c) and (d) are the processed results as a function of decreasing threshold. Because the internal operation of the circuit-operates in analog mode, the interconnect strength can be programmed to be continuously tunable. Hence, the threshold may be determined based on the situation of the circuit and the degree of satisfaction with regard to the treatment. This is quite different from pure digital operations.

As another example, a variable-threshold network capable of processing an inseparable logic function has been used to detect the edge of a binary image. Experimentally, the Laplace operator of four neighboring windows is used detect the edge of a binary image. In this method, when five pixels are identical, the output is 0; otherwise, it is 1. This is a linear inseparable logic function and cannot be completed by a single-threshold

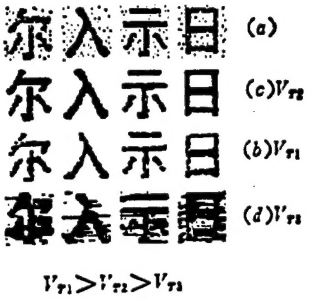


Figure 8. Experimental Result of Noise Cancellation of Binary Images

element. Despite the fact that this function is linear and inseparable in 5-dimensional space, it becomes a linear, separable logic function in a higher-dimensional space. Let $X_6 = X_1 + X_2 + X_3 + X_4 + X_5$ and use X_6 as the sixth input for the threshold element. The logic function by now becomes linear and separable and can be dealt with by a single-threshold element. Figure 9 shows the circuit used for edge detection using this method. Theoretically, this algorithm can also be accomplished by a variable-threshold logic network, however, this requires more neuro-chips and is therefore not used. Figure 10 shows the network treatment results, where (a) is the 34 x 34 pixel input binary image, and (b), (c) and (d) are the results obtained using different threshold values. From this edge detection example, transistor neural networks not only can be used to implement variable-threshold logic circuits but also common logic gates. Therefore, the circuit used to implement a logic function must be thoroughly considered in order to select the simplest one.

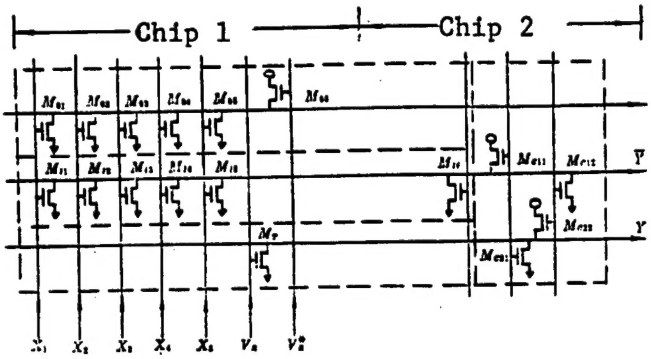


Figure 9. Edge Detection Circuit Using Four-Neighboring-Window Laplace Operator

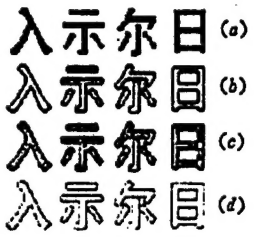


Figure 10. Experimental Results of Binary Image Edge Detection

V. Conclusions

A simple neuro-chip with variable interconnect weight has been designed and fabricated using 3 μ m floating-gate NMOS technology. This network behaves like a distributed neuron structure and can be cascaded into a large-scale network. The main part of the circuit is an 8 x 8 fully interconnected matrix, equivalent to an 8-neuron network. Weight storage is done by the floating-gate structure.

By connecting several floating-gate NMOS networks, a Hamming network was formed for numeral recognition and a variable-threshold logic network was created for noise cancellation of binary images. A binary and variable-threshold hybrid logic circuit was also formed to detect edges of binary images. All results are satisfactory. In these applications, the interconnect weights were electrically written and survived numerous write/erase cycles. This proves that the chip has a promising future and exhibits a high level of flexibility in various applications.

References

1. Atlas, L. E., Suzuki, Y., "Digital Systems for Artificial Neural Networks," IEEE CIRCUITS AND DEVICES MAGAZINE, 1989, 5: 20-24.

2. Mead, C., Ismail, M. (ed.), "Analog Implementation of Neural Systems," Kluwer Academic Publishers, Norwell, 1989.

3. Murray, A. F., Smith, A. V. W., IEEE J. SOLID-STATE CIRCUITS, 1988, SC-23:688-697.

4. Boahen, K. A., et al., IEEE TRANS., 1989 CAS-36: 747-755.

5. Wang Yang, "Programmable Transistor Neural Network Suitable for IC Technology," (Ph.D. Thesis), Qinghua University, Beijing, November 1991.

6. S. Satyanarayana, et al., ELECTRON. LETT., Vol 25 No 5, 1989, pp 302-303.

7. Wang Yang, Li Zhijian, and Shi Bingxue, "Behavior and Programming of Floating-Gate MOSFET as Synaptic Analog Storage," Proceedings of the Second National Neural Network Conference of China, Nanjing, 1991, pp 386-389.

8. Feldman, J. A., Ballard, D. H., "Connectionist Models and Their Properties," COGNITIVE SCIENCE, 1982, 6:205-254.

9. McCulloch, W. S., Pitts, W. A., "A Logical Calculus of the Ideas Immanent in Nervous Activity," BULL. MATH. BIOPHYS., 1943, 5:115-133.

Wang Yang graduated from the class of physics teachers in 1982 and earned his Master's degree in semiconductor physics and devices in 1986 from Harbin University, and received his Ph.D. degree in semiconductor devices and microelectronics from Qinghua University in 1992. He is doing post-doctoral research at Beijing University on realization of artificial neural networks via integrated circuits.

Li Zhijian graduated from the Physics Department of Zhejiang University in 1951 and received his associate doctor of science degree from the USSR in 1958. He is a professor at the Institute of Microelectronics of Qinghua University, a member of the Chinese Academy of Sciences, and vice chairman of the Chinese Society of Electronics. He has been engaged in the study of semiconductor devices and microelectronics technology throughout his career.

Shi Bingxue graduated from the Department of Radio Electronics of Qinghua University in 1959. He is a professor at Qinghua University. He has been involved in microelectronics research for some time and has published over 50 papers internationally. In recent years, his areas of interest have included analog integrated circuits and systems, artificial neural networks and their realization by means of VLSI.

Implementation of Fast Learning Algorithm for Multilayer Feed-Forward Neural Networks

93FE0193C Beijing DIANZI XUEBAO [ACTA ELECTRONICA SINICA] in Chinese
Vol 20 No 10, Oct 92 pp 61-68

[Article by Du Limin [2629 0448 3046] and Hou Ziqiang [0186 5261 1730] of the State Key Laboratory of Acoustics, Institute of Acoustics, the Chinese Academy of Sciences: "Implementation of Fast Learning Algorithm for Multilayer Feed-Forward Neural Networks"; MS received Mar 92, revised May 92]

[Excerpts]

Abstract

The optimum-learning-rate backpropagation algorithm is reviewed. The equations for computing the optimum learning rates of several commonly used networks are presented. Problems associated with the implementation of the algorithm are discussed. The speed of the algorithm is further illustrated by experimental simulations.

Key words: optimum learning rate, BP algorithm, multilayer neural network, prediction.

I. Introduction

In theory, it has been demonstrated that a layered feed-forward neural network can be employed to approximate any given continuous mapping f: $R^N \rightarrow R^M$ by properly selecting the topological architecture and the interconnect weight. This type of neural network plays an important role in artificial neural network research, especially in the recognition and classification of sensory signals such as voice and image, as well as in nonlinear signal processing applications such as adaptive filtering, adaptive control, and mapping approximation. This type of network model is often widely used as a critical part of a system or the entire system itself. The mapping created by such a network often becomes an important factor affecting the performance of the system. Nevertheless, we have not found an effective algorithm to automatically design the topological architecture and interconnect weight of a multilayer feed-forward network that provides a fixed mapping relation. In practice, it is converted to a two-stage experimental design optimization problem. This involves the selection of an architecture and using a learning algorithm to train the weight value to make the target function of the network reach an optimum or satisfactory level for this particular architecture. Then, the network architecture is changed and the training process of the interconnect weight is repeated. Finally, all the results are compared to choose the best combination of architecture and weight providing the optimum target function. The total time required to complete such a two-stage design process is equal to the number of network architecture chosen multiplied by the average time required to obtain the optimum or a satisfactory weight value for each architecture. Therefore, with a given network architecture and initial weight, it is of practical significance to reduce the learning time required to find the weight that provides a satisfactory target function value.

In practice, backpropagation (BP)[1] is the most popular algorithm for learning the weight of a multilayer feed-forward neural network. However, conventional BP is seriously impacted by the arbitrary constancy of the parameter selected. With a given network architecture and initial interconnect weight, one often needs to select different parameters for the algorithm and repeat the learning process to obtain a satisfactory solution. This not only significantly slows down the convergence rate of the design process but also diverts focus away from system architecture and functionality.

Specifically with reference to the difficulty caused by the arbitrary constancy in setting the parameter for back-propagation, a method to analyze the learning rate of an adaptive and improved BP algorithm is presented in reference 2. This method is formalized in reference 3 and general formulas are also provided. It essentially generates a second-order characteristic curve of the parameter (i.e., learning rate μ) along the direction of fastest decline of the target function by local linearization of the nonlinear processing elements in the network and this curve is used to locally approximate the target function. The second-order characteristic curve is completely determined by the current iteration gradient, the error of the processing element in the output layer and its perturbation. Its shape varies as the curvature of the target function at the point of interest changes. In every iteration, the optimum learning rate μ * always shifts the weight toward the minimum of the characteristic curve generated in the same iteration. The purpose of having the learning rate adapt according to the curvature of the target function of the network is to overcome the difficulty associated with the "tuning of the parameter of the algorithm," to improve the degree of the approximation of the mapping created by the network, and to speed up the network learning rate. This paper describes this algorithm and reviews its results. Problems associated with the implementation of the algorithm are also discussed. Specific formulas to calculate the optimum learning rates in frequently used networks are provided. The fast nature of the algorithm is also illustrated by way of simulation. (The derivation and proof of the algorithm is presented in reference 4.) [passage omitted]

IV. Computer Simulation Experiment

The objective of the experiment is to observe any improvement of the optimum-learning-rate BP algorithm over the conventional BP algorithm, in terms of learning speed and the ability of the network to create an approximation of the mapping desired. Furthermore, the effect of the initial weight on the results is also investigated. In addition, a comparison of the cost-to-benefit relation of computation is also made.

The objective of the experiment is to employ a single-hidden-layer feed-forward neural network to learn the Feigenbaum mapping produced by the following nonlinear iteration:

$$x(n+1) = rx(n)[1-x(n)], \quad n = 0, 1, 2... \qquad (22)$$

where the initial value $0 < x(0) < 1$, and r is a control parameter. In our experiment, let $x(0) = 0.0100004$ and $r = 4.0$. Equation (22) generates a general mixed-time sequence by way of iteration.

The purpose of using a layered feed-forward neural network to learn Feigenbaum mapping is to allow the weight of the network to begin learning from a random initial value. Eventually, it enables the network to produce an output x [caret over x] $(n + 1)$ in response to an input $x(n)$ in accordance with equation (22), i.e., absolute value of: x [caret over x] $(n + 1)$-$x(n + 1)^2$ is held to a minimum. This can also be viewed as a prediction of the time sequence generated by equation (22) by the layered feed-forward neural network. In the learning process, each experimental data block length is $T = 16$, and each block renews an input and ideal output pair.

The architecture of the network is chosen arbitrarily. In our experiment, we selected a hidden processing element comprised of six Sigmoid nonlinear response functions, a linear input processing element, a linear output processing element, and a non-zero threshold single-hidden-layer feed-forward network. The computation load for each iteration is shown in Table 1.

Table 1. Computations Required for Each Iteration for the (1 6 1) Network With T = 16

	Conventional BP algorithm	Optimum-learning-rate BP algorithm
Multiplication/division	853	1302
Addition/subtraction	512	830
Total	1365	2132

The experiment begins with two sets of initial weights. For each initial weight, let the conventional BP learning rates be 0.005, 0.01, 0.05 and 0.1 and initiate four corresponding learning processes. A total of eight learning processes are done for the two initial weights. For each learning process, the relative rms error of the network output (i.e., the ratio of the output error matrix mode to the ideal output matrix mode) is recorded as a function of the number of iterations involved. The results corresponding to two sets of initial weights are plotted in Figures 3(a) and 3(b), respectively. The two learning processes of the optimum-learning-rate algorithm associated with these two initial weights are also shown in Figures 3(a) and 3(b), as well.

For group A, as shown in Figure 3(a), the four learning curves controlled by the four learning rates essentially coincide with each other using the conventional BP algorithm. The convergence error is approximately -5 dB. (Experimentally, it showed no apparent improvement even when the number of iterations was increased to over 1,000.) In comparison, the learning curve of the optimum-learning-rate BP algorithm showed significant drop in about 100 iterations. In the vicinity of 300 iterations, it falls to a satisfactory level and the convergence error is under -20 dB. For group B, as shown in Figure 3(b), the four conventional BP curves are quite different. The 0.01-learning-rate curve has the best convergence rate and error. In the neighborhood of 500 iterations, its convergence error is about -40 dB. The optimum-learning-rate BP algorithm is still better than the four conventional BP learning curves. In approximately 500 iterations, the convergence error has declined to below -50 dB, corresponding to an improvement of mapping accuracy of 10 dB.

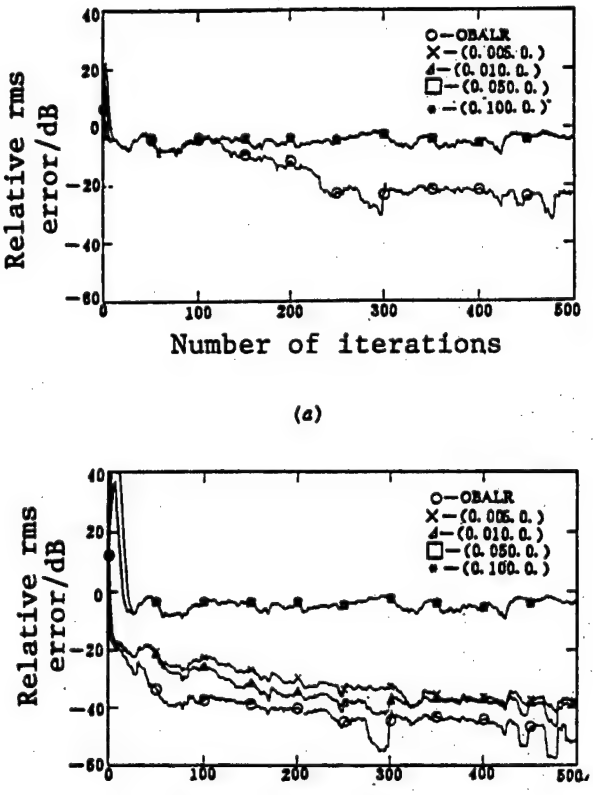


Figure 3. Comparison of Learning Process of Conventional BP Algorithm to Optimum-Learning-Rate BP Algorithm at Various Learning Rates; (a) and (b) are learning curves corresponding to initial values A and B.

If the number of arithmetic operations required to reach a specified convergence error is used to judge the convergence speed, then the optimum-learning-rate BP algorithm requires 5.3 x 10^5 arithmetic operations to reach -20 dB in group A. The conventional BP algorithm would only reach the -5 dB level after 2.7 x 10^6 operations. In group B, the conventional BP algorithm and optimum-learning-rate BP algorithm require 2.7 x 10^6 and 3.2 x 10^5 operations, respectively, to reach a convergence error of -40 dB. The optimum-learning-rate BP algorithm saves approximately an order of magnitude of arithmetic operations.

V. Discussion

Based on an analysis of the above experimental results, the following conclusions can be reached.

(1) When using the conventional BP algorithm to train a multilayer feed-forward network, it is very difficult to guess the proper parameters that produce a satisfactory convergence rate and convergence speed because there are so many factors affecting setting the parameters (such as the initial weight).

(2) Compared to conventional BP algorithm, the optimum-learning-rate BP algorithm significantly improves the convergence rate and error of the learning process. The experiment could not rule out the fact that there might be a parameter, out of numerous guesses, that would provide an even more satisfactory convergence rate and error using the conventional BP algorithm, as compared to the optimum-learning-rate BP algorithm. However, such a parameter could only be obtained by comparing the results of numerous learning processes after repetitive guessing and learning. The fine-tuning of the parameter is something we wish to avoid in artificial neural networks.

(3) In each iteration, learning-rate optimization takes approximately 56 percent of the amount of computation with a conventional BP algorithm. However, this additional computation load could greatly reduce the number of iterations and significantly improve the mapping accuracy to accelerate the overall learning process.

(4) The optimum-learning-rate BP algorithm still retains the characteristics of a gradient algorithm. Hence, the selection of the initial weight remains important. Usually, the initial value may be changed several times during training in order to improve the mapping characteristics generated by the network.

These four aspects as a whole are instrumental to the fast nature of the optimum-learning-rate BP algorithm.

References

1. D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning Internal Representations by Error Propagation," ICS Report, 1985: 8506.

2. Du Limin and Hou Ziqiang, "Fast Error Backpropagation Learning Algorithm—Adaptive Learning Rate Method," Proceedings of the First National Neural Network Conference, Beijing, 1990, pp 415-420.

3. Du Limin and Hou Ziqiang, "Optimum Adaptive Learning Algorithm for Error Backpropagation Network," Proceedings of the 1991 China Neural Network Conference, Nanjing, 1991, pp 267-269.

4. Du Limin, Hou Ziqiang, Li Qihu, "Optimum Block-Adaptive Learning Algorithm for Error Backpropagation Networks," IEEE TRANS. ON SIGNAL PROCESSING, 1992 (to be published).

5. V. Cuperman, A. Gersho, "Adpative Differential Vector Coding of Speech," Proc. of Globe Com82, 1982: 1092-1096.

6. W. B. Mikhael, F. H. Wu, "Fast Gradient Algorithms for Block FIR Adaptive Digital Filters," IEEE TRANS. ON CIRCUITS AND SYSTEMS, 1987, CAS-34: 1152-1160.

7. B. Widrow, M. E. Hoff, "Adaptive Switching Circuits," IRE WESCON Conv. Record, Part 4, 1960: 96-104.

8. C. Burrus, "Block Implementation of Digital Filters," IEEE TRANS., 1971, CT-18: 697-701.

9. R. Hecht-Hielsen, "The Theory of the Backpropagation Neural Network," Proc. of IJCNN '89, 1: 593-605.

Du Limin received his Bachelor of Science, Master of Engineering and Ph.D. in physics from Beijing University, graduate school of USTC, and Institute of Acoustics of the Chinese Academy of Sciences in 1983, 1987 and 1991, respectively. He is an Assistant Research Fellow at the State Key Laboratory of Acoustics, CAS Institute of Acoustics, and a member of IEEE. His areas of interest include artificial neural networks, fuzzy logic, Hidden Markov models, wavelet analysis, wavelet envelopes and their applications in voice recognition, and underwater acoustic signal recognition and processing.

Hou Ziqiang graduated from Beijing University in 1958 and is a Research Fellow at the CAS Institute of Acoustics, an advisor for Ph.D. candidates, and Secretary General of CAS. He is also Vice Chairman of the Society of Graphics and Images. He was engaged in hydroacoustic technology and signal processing in the 50s, 60s, and 70s and has received the National Science Award, Innovative Invention Award and CAS Technical Achievement Award. In 1980, he founded a high-tech company and since then has developed a number of advanced diagnostic systems such as an MRI-CT camera and a phase-controlled color Doppler B ultrasound diagnostic machine. He was given a First Class Technical Achievement award by CAS. He has authored several books, including "Digital Medical Diagnostic Imaging," "Sonar Signal Processing—Principles and Equipment," "Hydroacoustics," etc. His current areas of interest include fundamental theories in artificial neural networks, fuzzy logic, dimension theory, and signal processing theory and their applications in imaging, voice and hydroacoustic signal processing.

Optical Implementation of WTA Neural Network With Bipolar Neuron States

93FE0193D Beijing DIANZI XUEBAO [ACTA ELECTRONICA SINICA] in Chinese
Vol 20 No 10, Oct 92 pp 69-75

[Article by Shen Jinyuan [3947 6855 1254], Zhang Yanxin [1728 1693 3512], Wang Xuming [3769 6079 2494], and Mu Guoguang [3018 0948 0342] of the Institute of Modern Optics, Nankai University: "Optical Implementation of WTA Neural Network With Bipolar Neuron States"; supported by the National Natural Science Foundation and Doctoral Key Point Foundation, MS received Mar 92, revised Jun 92]

[Text]

Abstract

This paper presents a new scheme for the optical implementation of a bipolar WTA (Winner-Take-All) triple-layer neural network model and its experimental results. In the full bipolar mode, the input to the ith neuron in the middle layer consists of two parts. One is the input to that neuron in the unipolar mode and the other is one half of the reverse pattern corresponding to the stored pattern. The constant ½ can be implemented by dividing the input to each neuron into two equal parts, i.e., an opaque and a transparent part. Experimentally, the bipolar system was found to have higher storage capacity and addressability than a unipolar WTA system.

Key words: WTA neural network model, bipolar neural state, pattern recognition, multi-channel inner-product hologram.

I. Introduction

In recent years, neural networks have been a hot research topic worldwide. Many new models have been presented[1] and one of them is the WTA neural network model. It is believed that the WTA neural network is a mechanism in the brain.[2,3] The WTA neural network not only has a high storage capacity and addressability but also can implement independent association and mutual association.[4-6] Furthermore, the connection weights of WTA are 0, 1 (unipolar), or +1, -1 (bipolar). A bipolar WTA model has larger storage capacity and higher addressability than its unipolar counterpart. Nevertheless, because negative neuron inputs and interconnects are involved, it is somewhat difficult to implement optically. The implementation of bipolar connection has been reported in several studies.[7-10] However, there has been no report on the simultaneous implementation of bipolar neuron state and interconnection. Wang Xuming and Mu Guoguang proposed a simple scheme to implement a bipolar Hopfield model by adding an additional row of elements on the interconnect mask to take advantage of a preset distributive background.[11] I. Shariv used a birefringent crystal to split a polarized light beam into two orthogonally polarized beams to represent the positive and negative neuron states and implemented a bipolar triple-layer network in a double-pass system.[12] In this work, our original unipolar optoelectronic hybrid WTA pattern-recognition system[13] has been improved and an optical pattern-recognition system for the implementation of the bipolar WTA neural network is presented. When a stored pattern contains another stored pattern, a unipolar system cannot accurately recognize it. For example, when shi [0577] and wang [3769] are both stored patterns, the original unipolar system cannot accurately judge whether the input is shi or wang when the input is shi. It can be recognized by a bipolar system. Therefore, the storage capacity and addressability of the network is significantly increased.

II. Bipolar WTA Neural Network Model

1. WTA Neural Network Model

Figure 1 shows the architecture of the WTA neural network model. It is a triple-layer neural network with a WTA hidden layer.[4-6] Each neuron in the hidden layer corresponds to a stored pattern. The number of neurons is equal to the number of stored patterns M. The connection between the jth neuron in the hidden layer and the ith neuron in the input layer is W_{ij}. The interconnect between the ith and jth neurons in the hidden layer is T_{ij}. The connection between the jth neuron in the hidden layer and the ith neuron in the output layer is W'_{ij}. When W_{ij} is equal to W'_{ij}, the network implements homo-association based on content addressability. Otherwise, it implements hetero-association.

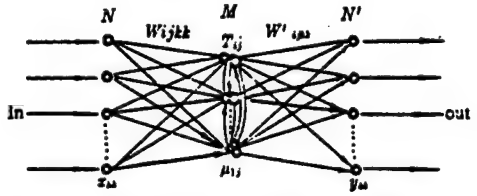


Figure 1. WTA Neural Network Model

The input pattern is the weighted summation of W_{ij} to determine the likelihood with every stored pattern. This likelihood is the input to the neuron in the hidden layer. When W_{ij} is the component V_i of the ith neuron corresponding to the jth stored pattern, the likelihood of the input pattern to the stored pattern is their inner product, i.e., the input to the jth neuron in the hidden layer is the inner product of the jth stored pattern and the input pattern. In the hidden layer, due to its connection T_{ij}, only the output from the neuron of the largest input is non-zero. The rest of them are all 0 and that completes the WTA operation. When the inner products between two or more patterns and the input pattern are equal, the network cannot properly recognize them. Its output will include all or none of them. Hence, one of the primary methods to raise the storage capacity and addressability of the model is to alter the weight W_{ij} to eliminate the possibility that more than one stored pattern has the same inner product with the input pattern. In the bipolar WTA model, $W_{ij} = X_i^j$, X_i^j is either +1 or -1. Compared to the unipolar WTA model ($W_{ij} = V_i^j$, V_i^j is either 1 or 0), it eliminates the problem where the system cannot accurately distinguish between two patterns when one pattern is contained in another. Therefore, the bipolar WTA model has larger storage capacity and better content addressability.

2. Bipolar Neuron and Interconnect Weight in WTA Model

The stored pattern in this work is a two-dimensional vector. The neuron in the hidden layer (hl) and the neuron in the input layer (jk) are connected by a four-dimensional tensor W_{jkhl}. X_{jk}^{hl} and X_{jk} represent the bipolar form of the hl stored pattern and the jk component of the input pattern, respectively. Let the interconnect W_{jkhl} be X_{jk}^{hl}. Then, the input to neuron hl of the hidden layer μ_{hl} is

$$\mu_{hl}=\sum_{k=1}^{K}\sum_{j=1}^{J} W_{jkhl}X_{jk}=\sum_{k=1}^{K}\sum_{j=1}^{J}X_{jk}^{hl}X_{jk} \quad (1)$$

where h = 1, 2, ...H and l = 1, 2, ...L represent the position of a stored pattern in the H x L matrix, H x L = M is the number of stored patterns. k = 1, 2, ...K, j = 1, 2, ...J and J x K = N represent that a stored pattern is a J x K matrix and N is the number of neurons in the stored pattern.

Any bipolar pattern (X_{jk}) can be expressed in terms of its unipolar format (V_{jk}):

$$X_{jk} = 2V_{jk} - 1 \quad (2)$$

Substituting equation (2) into (1), we have

$$\mu_{hl}=\sum_{k=1}^{K}\sum_{j=1}^{J}(2V_{jk}^{hl}-1)(2V_{jk}-1)$$

$$=4\sum_{k=1}^{K}\sum_{j=1}^{J}\left(V_{jk}^{hl}V_{jk}-\frac{1}{2}V_{jk}^{hl}-\frac{1}{2}V_{jk}\right)+N \quad (3)$$

The relation between a pattern (V_{jk}^{hl}) and its corresponding reverse pattern ($\bar{V}_{jk}^{hl}$) can be expressed as follows:

$$V_{jk}^{hl}=1-\bar{V}_{jk}^{hl} \quad (4)$$

$$\text{Hence,} \quad \sum_{j=1}^{J}\sum_{k=1}^{K}V_{jk}^{hl}=N-\sum_{j=1}^{J}\sum_{k=1}^{K}\bar{V}_{jk}^{hl} \quad (5)$$

From equation (3), we get

$$\mu_{hl}=4\sum_{k=1}^{K}\sum_{j=1}^{J}\left(V_{jk}^{hl}V_{jk}+\frac{1}{2}\bar{V}_{jk}^{hl}-\frac{1}{2}V_{jk}\right)-N \quad (6)$$

As far as a network with a given number of neurons and number of input-patterns (V_{jk}) is concerned,

$$C=N+2\sum_{j=1}^{J}\sum_{k=1}^{K}V_{jk} \quad (7)$$

is a constant. Hence, equation (6) can be written as:

$$\mu_{hl}=4\sum_{k=1}^{K}\sum_{j=1}^{J}\left(V_{jk}^{hl}V_{jk}+\frac{1}{2}\bar{V}_{jk}^{hl}\right)-C \quad (8)$$

We already know that each neuron in the hidden layer corresponds to a stored pattern and only the output of the neuron of the largest input value is not zero. Obviously, the output of the neuron in the hidden layer is only dependent upon the relative value of μ_{hl} and is independent of its absolute value. Hence, the constant C and the factor 4 in equation (8) can be omitted. Thus, μ_{hl} can be simplified as follows:

$$\mu_{kl} = \sum_{k=1}^{K} \sum_{j=1}^{J} \left(V_{jk}^{kl} V_{jk} + \frac{1}{2} \bar{V}_{jk}^{kl} \right) \qquad (9)$$

Based on equation (9), this bipolar WTA model may be implemented by connecting unipolar patterns. The input to the hidden layer of the bipolar neural network is comprised of two parts. One is the input to the corresponding unipolar neural network and the other is one-half of the sum of all components of the reversed unipolar stored pattern. The second part can be implemented by adding a translucent plate of identical arrangement to the stored pattern (as shown in Figure 2) next to the input pattern and by incorporating a reversed unipolar pattern in the interconnection (as shown in Figure 3). Cutting the transmittance to ½ is implemented by dividing a neuron into two parts; one has a transmittance of 1 and the other 0. Thus, the difficulty associated with the accurate control of ½ transmittance is avoided.

Figure 2. Input Pattern "+"

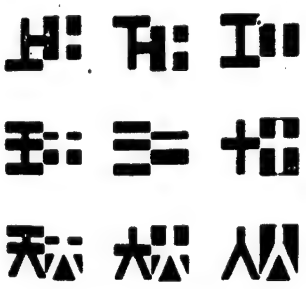


Figure 3. Stored and Reversed Stored Patterns of Interconnected Holographic Recording

The interconnect T_{ijkl} of neurons in the hidden layer is implemented through the use of an electronic network, as shown in Figure 4. It is dependent upon the maximum input of the neuron. In the iteration process, the neuron with the highest input is excited and provides an output of 1. Others are inhibited and provide an output 0.

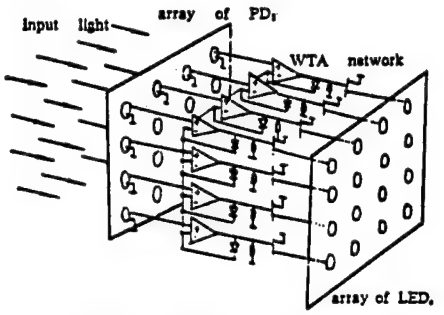


Figure 4. WTA Electronic Network

When the interconnection between the hidden layer and the neuron in the output layer, W'_{jkhl}, is equal to W_{jkhl}, the system proceeds with self-association. Otherwise, it carries out mutual association. When W'_{jkhl} is simultaneously chosen to be equal to and not equal to W_{jkhl}, the system performs self-association and mutual association at the same time.

III. Bipolar Multi-Channel Interconnection Holography

Because of advantages such as high storage capacity, solid space distribution and distributive storage of multiple three-dimensional objects, memory holography is widely used in optical neural networks.[14-16] These usually play the role of interconnection in an optical neural network. Figure 5 shows the optical configuration of the interconnected bipolar holographic system. P_1 is the input plane, and all the stored and reversed stored patterns are paired on P_1. Various stored patterns are separated in space. P_2 is the plane where a holographic interference plate is placed to record the interconnection. The distances between lens L_2 and P_1 and P_2 are d_1 and d_2, respectively. Furthermore, the following image formation relation is satisfied:

$$1/d_1 + 1/d_2 = 1/f \qquad (10)$$

where f is the focal length of L_2.

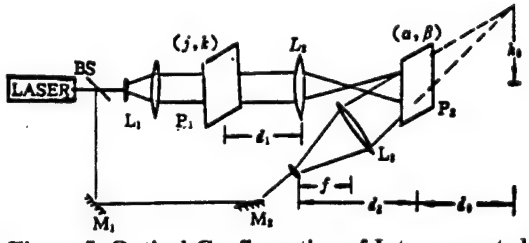


Figure 5. Optical Configuration of Interconnected Holography

Let us assume that the continuous format of the hlth stored pattern pair is $(V_{hl}(j-a_h, k-a_l) + \nabla_{hl}(j-a_h, \bar{k}-a_l))$, i.e., $V_{hl}(j-a_h, k-a_l)$ is the continuous format for the hlth stored pattern (V^{hl}_{jk}) and its coordinate on the input plane is (a_h, a_l). $\nabla_{hl}(j-\bar{a}_h, k-\bar{a}_l)$ is its reversed pattern and it is located at $(\bar{a}_h, \bar{a}_l)$ on P_1. Then, the complex optical field distribution on P_2 is

$$O_{hl}(\alpha,\beta)=c\{[V_{hl}(\alpha-a_h,\ \beta-a_l)+\bar{V}_{hl}(\alpha-\bar{a}_h,\ \beta-\bar{a}_l)] * h_{d1}(\alpha,\beta)\}T_2(\alpha,\beta) * h_{d2}(\alpha,\beta)$$
$$=c[V_{hl}\left(-\frac{d_1}{d_2}(\alpha+a_h),\ -\frac{d_1}{d_2}(\beta+a_l)\right)+V_{hl}\left(-\frac{d_1}{d_2}(\alpha+a_h),\ -\frac{d_1}{d_2}(\beta+a_l)\right)$$
$$\exp\left[\frac{ikd_1}{2fd_2}(\alpha^2+\beta^2)\right] \qquad (11)$$

Here, h_{d1} and h_{d2} are the pulse response functions of d_1 and d_2 with reference to free space, respectively. $T_2(x,y)$ represents the phase transformation introduced by lens L_2. * represents convolution and c is an exact function.

Then, the complex optical field of the input object on plane P_2 is

$$O(\alpha,\beta)=\sum_{h=1}^{H}\sum_{l=1}^{L}O_{hl}(\alpha,\beta) \qquad (12)$$

A convergent spherical wave is used as the reference. Its complex optical field distribution on P_2 is:

$$R(\alpha,\beta)=\exp\{-\frac{ik}{2d}[(\alpha-h_0)^2+\beta^2]\} \qquad (13)$$

The total complex optical field on plane P_2 is

$$O(\alpha,\beta)+R(\alpha,\beta)=\sum_{h=1}^{H}\sum_{l=1}^{L}O_{hl}(\alpha,\beta)+R(\alpha,\beta) \qquad (14)$$

The optical field is recorded holographically. After processing the exposed plate, the part of amplitude transmittance that is related to first-order diffraction is

$$t(\alpha,\beta)=O^*(\alpha,\beta)R(\alpha,\beta)$$
$$=c\sum_{h=1}^{H}\sum_{L=1}^{L}[V_{hl}(-\frac{d_1}{d_2}(\alpha+a_h),\ -\frac{d_1}{d_2}(\beta+a_l))$$
$$+\bar{V}_{hl}(-\frac{d_1}{d_2}(\alpha+\bar{a}_h),\ -\frac{d_1}{d_2}(\beta+\bar{a}_l))]$$
$$\exp\{-\frac{ik}{2d_0}[(\alpha-h_0)^2+\beta^2]-\frac{ikd_1}{2fd_2}(\alpha^2+\beta^2)\} \qquad (15)$$

The stored patterns in this work contain nine Chinese characters; their holograms are shown in Figure 3.

IV. Experimental Results

Figure 6 shows an optoelectronic WTA pattern-recognition system. P_1 is its input plane. A grating is placed at P_2 behind the imaging lens L_2. P_3 is the interconnected hologram. P_4 is the output plane of the holographic plate. V'(j,k) is the input to the input plane. It consists of a unipolar input pattern and a translucent pattern, as shown in Figure 2. Then, the complex optical field distribution on plane P_3 is

$$E(\alpha,\beta)=\{[V'(\alpha,\beta) * h_{d1}(\alpha,\beta)]T_2(\alpha,\beta)T_g(\alpha,\beta)\} * h_{d2}(\alpha,\beta)\,t(\alpha,\beta)$$
$$=c\sum_{h=1}^{H}\sum_{l=1}^{L}\exp\{-\frac{ik}{2d_0}[(\alpha-h_0)^2+\beta^2]\}\times$$
$$\{[V(-d_1(\alpha/d_2+a_h,-d_1)(\beta+a_l/d_2))V_{h1}(-d_1(\alpha+a_{h1}/d_2)-d_1(\beta+a_l/d_2))$$
$$\exp[ikd_1(a_h\alpha+a_l\beta)/d_2+1/2\exp[ikd_1(\bar{a}_h\,\alpha+\overline{a_l\beta})/d_{d2}]\times$$
$$\bar{V}_{hl}(-d_1(\alpha+\bar{a}_h/d_2),\ -d_1(\beta+\bar{a}_l/d_2))]\} \qquad (16)$$

where

$$T_g(\alpha,\beta)=1/4[1+\mathrm{Sgn}(\cos(\omega\alpha))(1+\mathrm{Sgn}(\omega\beta))] \qquad (17)$$

is the transmittance of the grating, $\omega = 2\pi a/\lambda d_2$, and a is the distance between various stored patterns. The complex optical field distribution at output plane P_4 of the connected associative storage device is

$$E'(\zeta,\eta) = E(\zeta,\eta) * h_{d0}(\zeta,\eta)$$
$$= c\sum_{h=1}^{H}\sum_{l=1}^{L}\iint d\alpha d\beta \exp\left[ik\left(\frac{d_1}{d_2}a'_h + \frac{\alpha - h_0}{d_0}\right)\zeta + \left(\frac{d_1}{d_2}a'_l + \frac{\beta}{d_0}\right)\eta\right]$$
$$\left[V(-d_1\alpha/d_2 + a_h, -d_1\beta/d_2 + a_l)\times\right.$$
$$V_{hl}(-d_1\alpha/d_2 + a_h, -d_1\beta/d_2 + a_l)$$
$$\left.+ 1/2\overline{V}_{hl}(-d_1\alpha/d_2 + \overline{a}_h, -d_1\beta/d_2 + \overline{a}_l)\right] \quad (18)$$

Obviously, the bipolar inner product of input pattern V(j,k) and output pattern $V_{hl}(j,k)$ can be obtained on plane P_4. It is located at $(h_0 + d_0d_1a'_h/d_2, d_0d_1a'_l)$, where $a'_h = (a_h + \bar{a}_h)/2 = ha$, $a'_l = (a_l + \bar{a}_l)/2 = la$. An optoelectronic triode is used to convert the intensity of the inner product to an electrical signal and this signal is transmitted to the WTA circuit in the hidden layer. By means of the interconnection effect, the winner takes all. Only the neuron with the largest input provides a high-voltage output. Others send a low-voltage output. This high-voltage signal is used to control a LED to illuminate the corresponding self-association or mutual association to thus provide associative recognition. Figure 7 [photograph not reproduced] shows the results. The use of bipolar connection and bipolar neuron input overcomes the problem wherein accurate identification is not possible with a unipolar WTA system when a pattern contains another pattern. The storage capacity and addressability of the system have also been improved.

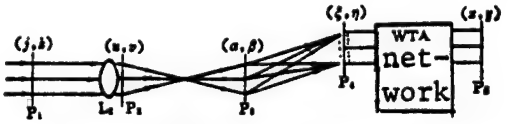


Figure 6. WTA Pattern Recognition System

V. Conclusions

Both theoretical analysis and experimental results illustrate that a bipolar WTA model has a larger storage capacity and higher addressability. It not only is capable of performing self-association but also mutual association. In addition, its illumination and gray scale are invariant. This is because the output of the neurons in the hidden layer is only dependent upon the relative, but not the absolute, value. If a pre-processor is placed in front of the system and the reference pattern is replaced by a corresponding invariant characteristic, then the system may be invariant with reference to inner and outer rotation, scale, and displacement.

The storage capacity of the system is primarily limited by the fabrication technique, such as the inhomogeneity of the multi-channel inner product holographic plate and optoelectronic devices.

References

1. R. P. Lippmann, "An Introduction to Computing With Neural Nets," IEEE TRANS., 1987, ASSP-35 (4): 4-22.

2. A. L. Yuille, et al., "A Winner-Take-All Mechanism Based on Presynaptic Inhibition Feedback," NEURAL COMPUTATION, 1989, 1 (3): 334-347.

3. W. F. Colmers, et al., "Presynaptic Action of Neuropeptide Y in Area CAl of Rat Hippocampal Slice," J. PHYSIOL., 1985, 383: 285-299.

4. H. H. Arsenault and B. Macukow, "Beyond Pattern Recognition With Neural Nets," in "Real-Time signal Processing for Industrial Applications," B. Javidi, ed., PROC. SPIE 960, 1989: 206-216.

5. H. H. Arsenault, "Neural Network Model for Fast Learning and Retrieval," OPT. ENG., 28(5): 506-512 (1989).

6. Zhang Yanxin, et al., "Simulation and Analysis of the Three WTA Models," PR&AI, Vol 5, No 1, 1992, pp 1-7.

7. D. Psaltis, et al., "Optical Information Processing Based on an Associative Memory Model of Neural Nets With Thresholding and Feedback," OPT. LETT., 1985 (10): 98-100.

8. T. W. Lu, et al., "Two-Dimensional Programmable Optical Neural Network," APPL. OPT., 1989, 28 (22): 4908-4913.

9. Ju-Seog Jang, et al., "Optical Implementation of the Hopfield Model for Two-Dimensional Associative Memory," OPT. LETT., 1988, 13 (3): 248-250.

10. Seok Ho Song, et al., "Properties of Holographic Associative Memory Prepared by the Polarization Encoding Process," APPL. OPT., 1988 (27): 3149-3154.

11. Xu-Ming Wang, et al., "Holographic Associative Memory With Bipolar Features," "Wave Propagation and Scattering in Varied Media," V. K. Varadan eds., PROC. SPIE 1558, 1991: 518-528.

12. I. Shariv, et al., "All-Optical Bipolar Neural Network With Polarization-Modulating Neurons," OPT. LETT., 1991, 16 (21): 1692-1694.

13. Y. X. Zhang, et al., "Optical/Electronic Hybrid Three-Layer Neural Network for Pattern Recognition," "Applications of Artiifical Neural Network," Steven K. Rogers ed., PROC. SPIE, 1469, 1991: 303-307.

14. Q. W. Song and Francis T. S. Yu, "Holographic Associative Memory System Using a Thresholding Microchannel Spatial Light Modulator," OPT. ENG., 1989, 28 (5): 533-536.

15. X. M. Wang, et al., "Optical Associative Memory Using an Orthogonalized Hologram," OPT. LETT., 1991, 16 (2): 100-102.

16. Eung Gi Paek, et al., "Optical Associative Memory Using Fourier Transform Holograms," OPT. ENG., 1987, (26): 428-433.

Shen Jinyuan was born in 1966 and graduated from the Department of Electronics at Nankai University with her Bachelor of Science degree in 1987. In 1990, she earned her Master of Science from the Institute of Modern Optics of Nankai University and is studying for her Ph.D. degree. Her areas of interest are optical information processing, pattern recognition and optoelectronic neural networks.

Zhang Yanxin is a professor and a doctoral adviser at Nankai University. He is also Associate Director of the Institute of Modern Optics. His research interests include infrared and visible detectors and instrumentation, optoelectronic information processing, optical neural networks and neural-network computers, optoelectronic devices and system, etc. Professor Zhang is a member of SPIE and INNS. He is also a member of the subcommittee on optical information processing and holography and the subcommittee on optoelectronics of the Chinese Society of Optics.

Wang Xuming was born in 1962 and is an Associate Research Fellow. He graduated from the undergraduate and graduate program from the Department of Physics at Nankai University in 1983 and 1986, respectively. He has been working at the Institute of Modern Optics since 1986. In 1990, he received his doctoral degree on the job. His primary research areas include optical pattern recognition, information processing and optical neural networks. He has published more than 20 papers in important technical journals and at international meetings worldwide.

Mu Guoguang is the President of Nankai University and the Director of the Institute of Modern Optics. He is also a professor and a doctoral adviser and a member of the technology committee of the Chinese Academy of Sciences. His areas of interest include applied optics, optical information processing and holography, optical pattern recognition and optical neural networks. He is the Vice Chairman of the Chinese Society of Optics and a Fellow of OSA and SPIE.

Aircraft Classification Under Affine Transformation Using Neural Network

93FE0193E Beijing DIANZI XUEBAO [ACTA ELECTRONICA SINICA] in Chinese
Vol 20 No 10, Oct 92 pp 76-81

[Article by Jin Qi [6855 1142] and Yan Pingfan [7051 1627 0416] of the Department of Automation, Qinghua University: "Classification of Aircraft Under Affine Transformation Using a Neural Network"; MS received Apr 92, revised Jun 92]

[Text]

Abstract

The Fourier descriptor method is extended to produce a set of invariants that are independent of the affine transformation. These invariants are used to train a triple-layer perceptron network for the identification and classification of aircraft. An accelerated learning algorithm is adopted to significantly reduce the learning time. Finally, the results of using such a neural network for identification and classification of aircraft and specification of noise tolerance are also presented.

Key words: affine transformation, invariant, neural network, backpropagation, classification.

I. Introduction

Identification of a two-dimensional pattern is an important subject. Usually, this type of problem is done by similarity transformation, i.e., translation, rotation, magnification or reduction. Intuitively, the shape remains invariant. When a plane pattern (e.g., an aircraft) is observed at different viewing angles, its shape cannot remain unchanged as a result of affine transformation. To this end, K. Arbter[1] proposed identifying the pattern by finding a set of invariants through Fourier transformation. This set of invariants is given in the form of complex numbers. In this work, a new method is presented to find these invariants. This set of invariants is expressed in terms of real numbers. It does not take phase into consideration in performing Fourier transformation. These invariants can be used to train a three-layer perceptron to identify and classify aircraft. One of the major problems in using a neural network is the slow training process. To this end, an accelerated learning algorithm is introduced to significantly speed up the learning process. Finally, the noise tolerance of this neural network classifier is analyzed.

II. Brief Introduction to Affine Transformation

Any three-dimensional motion of a rigid body can be treated as a rotation about an axis that passes through

the origin and translation along the three axes.[3] The equation of motion can be expressed as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \qquad (1)$$

where $(dx, dy, dz)^T$ is the translation, R is a 3 x 3 rotation matrix, and $(x, y, z)^T$ and $(x', y', z')^T$ represent the coordinates of a point on the body before and after the movement, respectively. Let us assume that the plane of projection is z = 1, then their projections are:

$$X' = x'/z', \; Y' = y'/z', \; X = x/z, \; Y = y/z \qquad (2)$$

If the aircraft is far away from the image plane and varies very slightly in the z direction, i.e., z' z, then the transformation on the projection plane can be expressed as follows:

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad (3)$$

Or, in vector format, it is:

$$W' = AW + B \qquad (4)$$

Where A is a 2 x 2 matrix and det(a) not equal to 0, and W', W and B are two-dimensional column vectors. This is affine transformation.

We know that if matrix A can be expressed as A = aU, where a is a constant and U is an orthogonal matrix, then equation (4) becomes a similarity transformation. Affine transformation does not place any requirement on A. Therefore, similarity transformation is a part of affine transformation. Affine transformation contains similarity transformation. Figure 1 shows a few examples of such transformations.

Figure 1. Explanation of Affine Transformation

III. Determination of Invariants

The key to performing a Fourier transformation is to locate a parameter that varies linearly with respect to an arbitrary affine transformation. Arbter[1,2] provided a parameter t:

$$t = \frac{1}{2}\int_c (x\,dy - y\,dx). \qquad (5)$$

where c is the silhouette of the plane pattern and proved that the following conditions exist when B = 0 in equation (4).

(1) The parameter varies linearly under affine transformation, i.e., if t' is the parameter corresponding to the pattern to be identified and t is the parameter for the reference pattern, then constants c and d exist to satisfy t' = c(t + d).

(2) This parameter is independent of the selection of the initial point on the curve.

When $B \neq 0$, the origin of the coordinate system can be moved to B. $B=\oint_D W dx dy \Big/ \oint_D dx dy$ and its accuracy can be proved as follows. If the reference pattern satisfies $\oint_D W dx dy \Big/ \oint_D dx dy = 0$, and a transformation $W' = AW + B$ is made to change the enclosed area from D to D', then

$$B' = \oint_{D'} W' dx dy \Big/ \oint_{D'} dx dy =$$

$$\oint_D (AW+B) dx dy \Big/ \oint_D dx dy = B.$$

Theorem 1. The translation in an affine translation can be determined by

$$B=\oint_D W dx dy \Big/ \oint_D dx dy \qquad (6)$$

If an arbitrary point on the edge is expressed as a vector function $W=\begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$, after performing Fourier transformation with respect to x(t) and y(t), a matrix of coefficients expressed in terms of Fourier series is obtained.

$$\begin{bmatrix} \cdots X_0 X_1 X_2 \cdots \\ \cdots Y_0 Y_1 Y_2 \cdots \end{bmatrix} \qquad (7)$$

In equation (4), when B = 0, the discussion can be carried out in two cases.

1) W'(t) = AW(t) and Fourier transformation is done with respect to both sides:

$$\begin{bmatrix} X_n^{r\prime} + iX_n^{i\prime} \\ Y_n^{r\prime} + iY_n^{i\prime} \end{bmatrix} = A \begin{bmatrix} X_n^r + iX_n^i \\ Y_n^r + iY_n^i \end{bmatrix}$$

where X_n^r and X_n^i represent the real and imaginary part of the nth-order coefficient X_n. Based on the definition of equal complex numbers, we have

$$\begin{bmatrix} X_n^{r\prime} \\ Y_n^{r\prime} \end{bmatrix} = A \begin{bmatrix} X_n^r \\ Y_n^r \end{bmatrix}, \qquad \begin{bmatrix} X_n^{i\prime} \\ Y_n^{i\prime} \end{bmatrix} = A \begin{bmatrix} X_n^i \\ Y_n^i \end{bmatrix}$$

Therefore, $\begin{vmatrix} X_n^{r\prime} & X_n^{i\prime} \\ Y_n^{r\prime} & Y_n^{i\prime} \end{vmatrix} = |A| \begin{vmatrix} X_n^r & X_n^i \\ Y_n^r & Y_n^i \end{vmatrix}$

2) $W'(t_1) = W(t)$ and $t_1 = C(t + d)$. After performing Fourier transformation with respect to both sides, we have

$$\begin{bmatrix} X_n^{r\prime} + iX_n^{i\prime} \\ Y_n^{r\prime} + iY_n^{i\prime} \end{bmatrix} = e^{inb} \begin{bmatrix} X_n^r + iX_n^i \\ Y_n^r + iY_n^i \end{bmatrix}$$

where b is a constant.

Let $X_n = |X_n| e^{i\phi_x}$, $Y_n = |Y_n| e^{i\phi_y}$,

then,

$$\begin{vmatrix} X_n^r & X_n^i \\ Y_n^r & Y_n^i \end{vmatrix} = |X_n|\,|Y_n| \begin{vmatrix} \cos\phi_x & \sin\phi_x \\ \cos\phi_y & \sin\phi_y \end{vmatrix} =$$

$$|X_n||Y_n| \sin(\phi_y - \phi_x)$$

$$\begin{vmatrix} X_n^{r\prime} & X_n^{i\prime} \\ Y_n^{r\prime} & Y_n^{i\prime} \end{vmatrix} =$$

$$|X_n||Y_n| \begin{vmatrix} \cos(\phi_x + nb) & \sin(\phi_x + nb) \\ \cos(\phi_y + nb) & \sin(\phi_y + nb) \end{vmatrix} =$$

$$|X_n||Y_n| \sin(\phi_y - \phi_x)$$

Combining these two cases, we get:

Theorem 2. For an arbitrary affine transformation

$$E_n = \begin{vmatrix} X_n^r & X_n^i \\ Y_n^r & Y_n^i \end{vmatrix}$$

is a set of invariants.

Figure 2 shows six aircraft models to be identified. Figure 3 shows eight different configurations of model D by means of affine transformation. Table 1 lists the invariants determined by the method described in this work.

Table 1. Invariants Determined Corresponding to Aircraft Shown in Figure 3

A	-2.11	0.385	-2.31	0.338	0.520	-.813	-0.401	0.171	-0.163	-0.312
B	-2.13	-0.90	-1.84	0.293	0.121	-.792	-0.132	0.159	-0.332	-0.107
C	-2.12	0.361	-2.30	0.363	0.499	-.843	-0.356	0.155	-0.228	-0.293
D	-2.25	-0.23	-2.26	0.325	0.574	-.927	-0.188	0.254	-0.347	-0.206
E	-2.19	0.241	-2.30	0.366	0.562	-.877	-0.283	0.220	-0.262	-0.251
F	-2.22	0.245	-2.29	0.358	0.546	-.886	-0.302	0.225	-0.276	-0.265
G	-2.15	0.306	-2.30	0.355	0.529	-.846	-0.346	0.194	-0.234	-0.277
H	-2.20	0.165	-2.29	0.342	0.531	-.910	-0.241	0.201	-0.317	-0.224

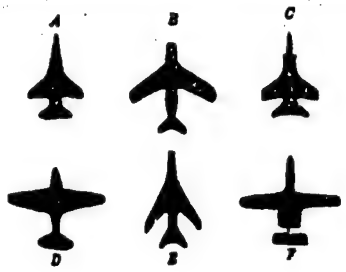


Figure 2. Six Aircraft Models Used

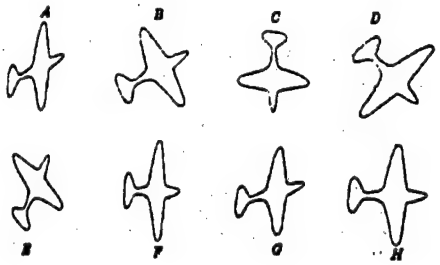


Figure 3. Configurations of Model D Obtained by Affine

IV. Neural Network Classifier

Several neural network models are used as classifiers. The most commonly used network is a multilayer perceptron. A multilayer perceptron is usually trained by backpropagation (BP). Let us assume that W_{ij} represents the connection weight between neuron i and neuron j in the subsequent layer. If d_j and y_j represent the ideal and actual output of an output node, respectively, then the overall error caused by a specific pattern can be expressed as follows:

$$E_p = \sum_j (d_j - y_j)^2 \qquad (8)$$

The weight correction process in the conventional BP method can be described as follows:

$$\Delta W_{ij} = -\eta \frac{\partial E_p}{\partial W_{ij}} \qquad (9)$$

where η is the learning rate. Weight correction is an iterative process up until the convergence condition absolute value of: $\Delta W_{ij} < c$ absolute value of: W_{ij} is met. As for the selection of η, usually a certain value is given before the learning process begins. If the converging process is slow, then η is gradually increased. If η is found to be too large, even causing oscillation, then η is gradually reduced. There are no specific methods to raise or lower η. Instead, one relies on experience. Usually η is chosen to be a constant. In this work, $\eta = 0.01$. We found that when the algorithm approaches its steady state, its convergence rate becomes very slow; this is because the weight correction method described earlier, i.e., the fastest fall method, is decreasing linearly. We also know that the speed of the Newton method that corrects the weight according to

$$\Delta W_{ij} = -\partial E_p / \sum_{ij} \left(\frac{\partial E_p}{\partial W_{ij}} \right)^2$$

decreases by the square power. However, its disadvantage is that each step might be so wide as to cause oscillation. Usually, the scheme is to use the fastest fall method to adjust it to near its steady state and then use the Newton method to correct the weight. The number of iterations can be drastically reduced. It is of great significance to choose the right η in order to minimize the number of iterations required. A method has been developed to dynamically calculate η.[4] Assume

$$\alpha = E_p \Big/ \sum_{ij} \left(\frac{\partial E_p}{\partial W_{ij}} \right)^2 \qquad (10)$$

then η can be determined as follows:

if α not equal to 0, then $\eta = \alpha$ if $\alpha < 20$; 20 otherwise,

if $\alpha = 0$, then $\eta = 0$.

After implementing this dynamic calculation of η in this work, the number of iterations was reduced from 900 to approximately 60.

V. Experimental Results

In this work, a three-layer perceptron is chosen as the neural network classifier. This three-layer perceptron has 15 input nodes and 6 output nodes. The number of nodes in the hidden layer is adjustable. We picked a range of 10 to 50. For the six aircraft shown in Figure 2, 15 different configurations are produced for each model by affine transformation. A total of 90 input samples are used to train this three-layer perceptron. In this work, the noise tolerance of this multilayer perceptron was also investigated. For each aircraft model, a group of noise-superimposed images was produced to test the accuracy rate of this classifier. Assuming that each edge point can move arbitrarily within an L x L window centered around this point, the noise rises as L increases. For instance, Figure 4, from left to right, shows the configurations as the noise level increases from 1 to 5. At every noise level L, 15 test patterns are generated for each aircraft. A total of 90 noise-superimposed images are used to test the accuracy of the network for identification of the aircraft. Figure 5 shows the error rate as a function of noise level L. It was found that even the network trained with noise-free specimens must also have a high resistance against noise. The reason is that it plays an important role in determining some of the invariants and a secondary role in the remaining invariants. As the noise level rises, the impact on the primary invariants is far less than that on the secondary ones. The three-layer perceptron can depend on the primary invariant to perform classification. Finally, noise-superimposed images were used to train this three-layer network and we found that it had a better noise tolerance.

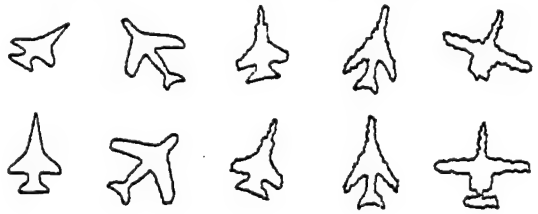


Figure 4. Noise-Superimposed Models (Noise Level 1 to 5 from left to right)

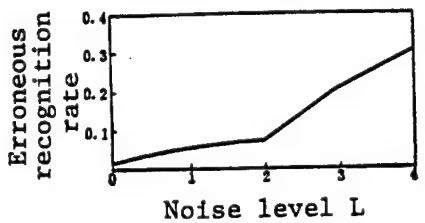


Figure 5. Noise Level vs. Identification Error

VI. Conclusions

An aircraft identification method involving affine transformation is presented. The mode recognition process is completed by BP network. This method is applicable to any plane pattern with a continuous and smooth border. The accelerated algorithm introduced can drastically reduce the computing time. It is general in nature and can be extended to many neural network learning processes.

References

1. K. Arbter, W. E. Snyder, "Application of Affine-Invariant Fourier Descriptors to Recognition of 3-D Objects," IEEE TRANS., July 1990, PRMI-12 (7).

2. K. Arbter, "Affine-Invariant Fourier Descriptors," in "From Pixels to Features," Amsterdam, The Netherlands: Elsevier Science, 1989.

3. T. S. Huang, R. Y. Tasi, "Image Sequence Analysis: Motion Estimation," in "Image Sequence Analysis," T. S. Huang (Eds.), Berlin: Springer-Verlag, New York: Heidelberg, 1981.

4. J. Schmidhuber, "Accelerated Learning in Backpropagation Nets," in "Connectionism in Perspective," R. Pfeifer et al. (Eds.), North-Holland: Elsevier Science Publishers B. V., 1989.

Jin Qi is a Ph.D. candidate in the Department of Automation at Qinghua University. His major research areas include computer vision, classification, image processing, artificial neural networks and their applications.

Yan Pingfan is a Ph.D. adviser and professor in the Department of Automation at Qinghua University. His current areas of interest include computer vision and artificial neural networks and their applications.

Neural Net DTW Architecture in Speech Recognition

93FE0193F Beijing DIANZI XUEBAO [ACTA ELECTRONICA SINICA] in Chinese
Vol 20 No 10, Oct 92 pp 82-87

[Article by Li Haizhou [2621 3189 3166] and Xu Bingzheng [1776 4426 6927] of South China University of Technology, Guangzhou: "A Neural Net DTW Architecture in Speech Recognition"; supported by the National Natural Science Foundation, MS received Mar 92, revised May 92]

[Text] **Abstract**

This paper presents a dynamic-programming-based time normalization method to implement a DTW neural network. DTW (dynamic time warping) is one of the most effective methods for spoken word recognition. It is

very robust and provides the highest recognition rate possible. However, it takes too much computing time. Unless with special hardware, its implementation is limited by time. In this work, all the computation is carried out by two recurrent subnets and a memory layer. This method demonstrates the superiority of the hard-wired architecture. It offers a new strategy for the implementation of DTW by hardware.

Key words: neural network, speech processing, pattern recognition.

I. Introduction

It is well known that because voice signals are highly random, speech patterns are often nonlinearly distorted along the time axis. How to resolve this problem, i.e., the time normalization problem, is an important issue in speech recognition. DTW is a nonlinear time normalization algorithm. Through the normalization of two input voice signals, their acoustic similarity can be maximized and their cumulative distance is minimized.

When using DTW to recognize an isolated voice signal, the unknown templates are compared one by one with the reference templates until a reference template with the least cumulative distance is found and this template is the result.

As we know, an effective algorithm can produce satisfactory results with the appropriate hardware. Recent studies[2,3] show that the neural network is an important technique to accomplish optimization. In this paper, we will demonstrate the use of a neural network to solve the dynamic normalization problem. Furthermore, an architecture is provided for the implementation of this neural network with optics and semiconductors.

The second part of this paper briefly discusses the DTW algorithm. The third part shows a neural network architecture to implement this algorithm. Finally, the efficacy of this method is verified with a real example.

II. Dynamic Time Normalization Algorithm

Speech can be expressed in terms of proper parameters. We use Eigen vectors to express voice signals:

$$A = a_1, a_2, \ldots a_i, \ldots a_I, \qquad B = b_1, b_2, \ldots b_j, \ldots b_J \qquad (1)$$

Now, let us consider how to eliminate the time difference between these two speech patterns. Consider the I-J plane shown in Figure 1. The I axis and J axis represent the unknown and reference pattern, respectively. The time difference between these two patterns may be expressed by a point sequence c = (i, j):

$$F = c(1), c(2), \ldots, c(k) \qquad (2)$$

where

$$c(k) = (i(k), j(k)) \qquad (3)$$

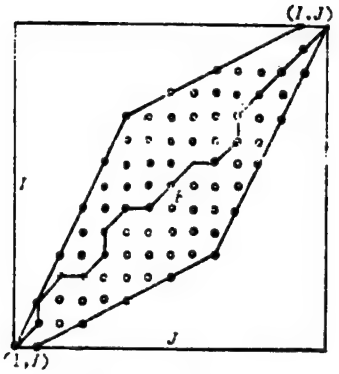


Figure 1. Limiting Window for Normalization on I-J Plane and Normalization Function

F is known as the normalization function. It is used to complete pattern matching from A to B. When there is no time difference between the two input speech patterns, the normalization function is the straight diagonal line i = j. As the time difference increases, the normalization function deviates farther from the straight line. In order to determine the difference between Eigen vector a_i and b_j, let us define local distance (LD) as follows:

$$d(c) = d(i, j) = \| a_i - b_j \| \qquad (4)$$

The weighted sum of LD is defined as the general distance (GD)

$$E(F) = \sum_{k=1}^{K} d(c(k)) \quad W(k) \qquad (5)$$

where W(k) is the weight coefficient which in conjunction with the slope constraint[1] controls the normalization function. When F is determined, E(F) is minimized. Correspondingly, the time difference between two matching specimens is adjusted. The time-normalized distance between patterns A and B is defined as follows:

$$D(A, B) = \min\left[\frac{\sum_{k=1}^{K} d(c(k)) \quad W(k)}{\sum_{k=1}^{K} W(k)}\right] \qquad (6)$$

where $\sum_{k=1}^{K} W(k)$ in the denominator is used to compensate for the effect of the weight on F.

Based on unique features associated with speech, a number of constraints such as monotony and continuous boundary of variation of speech parameters[1] are imposed on the normalization function. Usually, the

search area of the normalization function is limited to a window on the I-J plane, called the limiting window, as shown in Figure 1.

During the minimization process for D(A, B), the GD for every point shown in Figure 1 is determined. They may become a point in the point sequence of the normalization function F. The following iteration equation is used to optimize D(A, B).

$$g(c(k))=\min_{c(k-1)}[g(c(k-1))+d(c(k))W(k)] \tag{7}$$

In Figure 1, from left to right and from bottom to top, the partial general distance (PGD) is calculated step by step. In the equation, c(k - 1) includes all matching points in the previous step. We will discuss how to implement this process with two recurrent neural subnets in the following. In this case, every point is linked by code. For the problem under investigation in this work, equation (7) may be written as:

$$g(i,\ j)=\min\begin{bmatrix} g(i,\ j-1)+d(i,\ j) \\ g(i-1,\ j-1)+2d(i,\ j) \\ g(i-1,\ j)+d(i,\ j) \end{bmatrix} \tag{8}$$

Here, $\Sigma W(k) = I + J$. When the computation reaches the last point (I, J), D(A, B) can be optimized and the value is g(I, J)/(I + J).

If an optimization strategy table is used to record the local optimization path employed to derive the current PGD, then the normalization function F may search in the reverse direction.

III. Neural-Network-Guided Computing Architecture

In the optimization process, a recurrent chain table is created for all states associated with axes I and J. Each state is coded with a binary state vector Y in a recurrent network, i.e., the so-called network layer I and network layer J of the guiding network. For I and J, they are coded with M_1 and M_2 neurons in the form of (unordered set: -1, 1). Furthermore, it satisfies 2^M1 [1 is subscript for M] > I, 2^M2 [2 is subscript for M] > J. In Figure 1, to make it convenient to define the position of a point, I and J are coded and the two codes are combined into a state vector Y. Since the number of neurons grows logarithmically with respect to I and J, the scale of the network is suited for the matching of a large number of samples.

A state is used to describe a point with reference to guiding layer network I and J. A converter is used to match the binary state vector with the corresponding PGD and BTP (backward tracking pointer). This converter is implemented by the help of a forward network layer called the memory layer.

When the optimization process is completed, if necessary, it is possible to construct the optimal normalization path F by using the associative memory BTP. The tracking begins from the last node; the normalization function F may be reconstructed by connecting the nodes pointed out by BTP neurons.

1. Guiding Layer

This layer contains two recurrent neural subnets comprised of M_1 and M_2 fully connected high-level neurons,[2] respectively. We have $R = M_1 + M_2$. The state of the neuron in this layer is expressed as S = unordered set: Y^1, Y^2, ..., Y^2R [R is superscript for 2], where $Y \equiv (Y_1, Y_2, ..., Y_T) \varepsilon$ unordered set: -1, 1^T. Assuming y and y' are the past and present state, then the recurrent process can be described as follows:

$$y'_h(y)=W_h\eta(y)=\sum_{\alpha=00\cdots0}^{11\cdots1} W_h^{\alpha}\eta_{\alpha}(y) \tag{9}$$

The summation covers the entire M bits of the binary series $\alpha = \alpha_1\alpha_2...\alpha_M \varepsilon$ unordered set: 0, 1^M. We use $W_h = (W_h^{00...0}, W_h^{00...1}, ..., W_h^{11...1})$ to express the 2^M dimensional higher-order weight vector for neuron h. Then, by using $\eta \equiv (\eta_{00..0}, \eta_{00..1}, ..., \eta_{11..1}) \varepsilon$ unordered set: -1, 1^2M [M is superscript for 2] and

$$\eta_{\alpha}(y)=\prod_{h=1}^{M} y_h^{\alpha_h} \tag{10}$$

the state vector y may be expanded into a 2^M dimensional state vector. The hth bit of α in the equation provides the power of y_h. From the above, the architecture[3] provided by equations (9) and (10) is sufficient to produce any reflection of y': unordered set: -1, $1^M \rightarrow$ unordered set: -1, 1^M. The learning process of the recurrent weight is:

$$W_h=\frac{1}{2^M}\sum_{k=1}^{2^M} y_h'(y^k)\eta(y^k) \tag{11}$$

The state evolves in the recurrent subnets to create the node sequence shown in the chain table. In order to establish such two recurrent subnets, consider the following finite sequence $y^1 \rightarrow y^2 \rightarrow ... \rightarrow y^{kn}$; $k_i \varepsilon$ unordered set: 1, ..., 2^M. As discussed in the previous section, all neuron states are linked from bottom to top along the I axis and from left to right along the J axis when the system is initialized. Based on equation (9), a new state is totally determined by the state at the preceding moment. In addition, it is defined that y^1 immediately follows y^2M [M is superscript for 2] to form a recurrent chain.

In this network, to increase or decrease the number of states is a simple operation. If the present state y^{kj} is to be deleted, it can be accomplished by breaking the links established by equation (11) that connect the present state to the previous state and the present state to the future state.

the chain can be reconnected.

$$\Delta W_{\lambda} = -\frac{1}{2^{M}}\left(y_{\lambda}{}^{k}{}_{j+1}\eta(y^{k}{}_{j}) + y_{\lambda}{}^{k}{}_{j}\eta(y^{k}{}_{j-1})\right) \quad (12)$$

Then, according to the following equation

$$\Delta W_{\lambda} = \frac{1}{2^{M}} y_{\lambda}{}^{k}{}_{j+1}\eta(y^{k}{}_{j-1}) \quad (13)$$

This simple delete operation makes it easy to reconnect the chain in this network. When a new state is added to the chain table, it is implemented by using equations (12) and (13). The difference is that equation (13) must connect the previous state to the new state and the new state to the future state.

2. Memory Layer

In order to perform digital computing and BTP recording, a forward network is introduced to correspond all states in the first recurrent subnet with all interconnected LD, PGD and BTP. It is called the memory layer.

This network operates in a manner similar to the first layer. The system R has a binary input, but only outputs two real numbers; one is LD which later upgrades to PGD and the other is BTP. Therefore, the layer has two weight vectors, W_d and W_p. The learning process for the reflection of the state vector onto LD follows the same manner as expressed by equation (11).

$$W_{d} = \frac{1}{2^{R}}\sum_{k=1}^{2^{R}} Id(y^{k})\eta(y^{k}) \quad (14)$$

In the DTW treatment in this work, every node in the chain table has three local paths to select. After choosing the optimal local path based on equation (8), the minimum PGD is obtained. In addition, the previous point before the optimal path is denoted for backward search. This can be accomplished by using PGD to upgrade the LD neuron. This upgrade means that the weight W_d is updated according to the following:

$$\Delta W_{d} = \frac{1}{2^{R}}\left(pgd(y^{k}) - ld(y^{k})\right)\eta(y^{k}) \quad (15)$$

Furthermore, the BTP neuron is denoted by unordered set: -1, 0, 1 and each value represents a possible local path. We have

$$W_{p} = \frac{1}{2^{R}}\sum_{k=1}^{2^{R}} btp(y^{k})\eta(y^{k})$$

The BTP neuron may be initialized by training the weight so that the output of every column of the I-J plane from bottom to top is -1. When a point is located outside the limiting window, its output is 1. In other situations, its output is 0.

3. Auxiliary Control Unit

Figure 2 shows the complete network architecture. In the local path processing module, some control units are used.

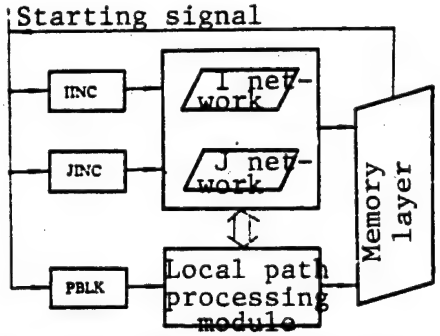


Figure 2. Flowchart of Network Control

The comparator COM is used to make a determination on the result produced by equation (8). Corresponding to the three input values, the comparator has two different outputs, the minimum g(i, j) and BTP marker. The output of the comparator simultaneously trips the upgrade of the LD neuron and the recording of the BTP marker.

An accumulator is placed in front of the comparator to generate the three possible PGD values for comparison.

The addresser ADDR monitors and records the possible initial addresses of local paths in subnets I and J. The unit is comprised of registers CUR to store the current state, and PRE1, PRE2 and PRE3 for storage of previous states (i, j-1), (i-1, j-1) and (i-1, j), respectively. For a given state (i, j), it looks up the address for state (i, j-1) by keeping subnet I invariant and evolving the J subnet J-1 times. Similarly, the addressing of (i-1, j-1) and (i-1, j) is done in the same fashion.

A number of triggers are employed in the system. They are added to the BTP neurons to guide the computing process.

Trigger SIG reports the completion of the normalization process, i.e., the optimization of F, and triggers the backward search.

Triggers IINC, JINC and PBLK control the operations in the evolution and processing modules of subnets I and J.

IV. Speech Recognition Experiment Using Isolated Characters

Speech signals are converted to digital signals by a 22 kHz A/D converter. The speech waveform is preprocessed with a partially overlapping Hamming window. The signal window is 22 milliseconds wide. Eigenvectors are separated after passing through a 20-channel narrow-band filter array.[4] The speech databank is comprised of 10 single-syllable words of the numbers 0-9 in Cantonese. The reference samples are provided by 20 males and 20 females. The test speech samples are taken from five females and five males other than those providing the referencing samples.

The Euler (?) distance between the two vectors is used to initialize the local distance LD of the network. The memory layer is initialized according to equation (14). The chain table of all the states is initialized prior to performing matching in subnet I and J independently. For a given vector pair to be matched (A, B), according to equations (12, 13), states y^{I+1}, ..., y^2M_1 [M_1 is superscript for 2] in net I are deleted from the subnet. State y' is connected to y^I to make the subnet recurrent. To determine the position of y^{I-1} from y^I is equivalent to have the subnet go through I-1 evolutions. Similarly, extraneous states in network J may be deleted accordingly.

Starting from state (1, 1), based on the description in Section 3, the outputs of the BTP neurons are sequentially read by triggers IINC, JINC and PBLK to determine the interconnection between states. Then, BTP is renewed to denote the optimal local path. In addition, the LD neuron is upgraded to the PGD value. When the BTP neuron output is 0 or 1, trigger IINC trips network I to evolve. When the BTP output is -1, trigger JINC trips network J to evolve. When the BTP output is 0, trigger PBLK trips the processing module into operation. When SIG indicates that the search is over, the general distance g(I, J) is compared to the minimum general distance MGD that is already stored in a register. If it is less than the MGD, then the corresponding reference sample index (ID) is replaced by the new word. Once the sample to be identified is compared with all the reference samples, the recognition process is completed. Figure 2 shows the control flowchart using neural-network-guided computation for speech recognition.

The optimal normalization function F can be denoted using the BTP neurons. Working just as ADDR in the processing module, the optimal path can be backtracked step by step. It is then linked together until reaching the starting point (1, 1).

It was found experimentally that the recognition rate for isolated words spoken by an unspecified person is as high as 99.0 percent. This is comparable to that of any conventional algorithm.[4]

V. Conclusions

This experiment points out a way to implement a conventional DTW technique by using a hardware network. This finding is of critical significance. As we know, a hard-wired architecture will undoubtedly increase the computing speed to reach optimization. As for the neural network presented in this work, the LD may be computed in parallel before optimization. All these advantages, on top of the apparent superiority of the parallel architecture of the neural network, make the implementation of DTW more effective.

References

1. H. Sakoe, S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition," IEEE TRANS., 1978, ASSP-26(1): 43-49.

2. G. Fahner, "A Higher-Order Neuron That Performs Arbitrary Boolean Functions," PROC. IEEE IJCN-N'90, San Diego, 1990, 3: 193-197.

3. G. Fahner, "A Neural Net That Manages a Hierarchical Spatial Data Base," ICANN-91, Helsinki, 1991, 811-817.

4. H. Li, et al., "Hidden Markov Models in Terms of a Mixture of Gaussian Phonetic States," JOURNAL OF COMPUTER PROCESSING OF CHINESE & ORIENTAL LANGUAGES (Canada), 1991, 5(2): 193-200.

Li Haizhou was born in 1964 and received his bachelor's and master's degrees in engineering from South China University of Technology in 1984 and 1987, respectively. In 1990, he earned his Ph.D. degree from Hong Kong University. Since 1990, he has been working at the Institute of Radio Technology of South China University of Technology as a lecturer and assistant professor. He is a member of IEEE and INNS. His research areas include digital signal processing, neural networks and pattern recognition. His major accomplishments include a book entitled "New Theory in Voice Signal Processing" and a patent on a "Voice-Controlled Chinese Typewriter."

Xu Bingzheng was born in 1925. He received his bachelor's degree (in electrical engineering) from Xiamen University in 1946 and his master's degree (in electronic engineering) from Lingnan University in 1950. He worked at South China University as a lecturer, associate professor, professor, chairman of the department of radio technology, and assistant president. He is Director of the Institute of Radio Technology and Automatic Control of South China University of Technology, Ph.D. advisor, board member and fellow of the China Electronics Society, and an IEEE senior member. He has devoted long periods of time to the study of information theory and signal processing. His major publications include "Signal Analysis and Related Technology" and "Theory of Digital Communications."

Approximate Logic for Neural Expert System

93FE0193G Beijing DIANZI XUEBAO [ACTA ELECTRONICA SINICA] in Chinese
Vol 20 No 10, Oct 92 pp 88-93

[Article by Hu Hong [5170 1347] and Shi Zhongzhi [0670 1813 2784] of the National Center for Intelligent Computing Systems, Institute of Computer Technology of the Chinese Academy of Sciences: "Approximate Logic for Neural Expert System"; MS received Mar 92, revised Jun 92]

[Text]

Abstract

This paper presents an approximate logic system. Not only the logic value of the system is fuzzy but also the logic operator can be fuzzy as well. This approximate logic system is very suitable for a neural network. A neural network model is established based on this concept. It is capable of learning and storing knowledge. On the basis of this neural net, a multi-expert opinion synthesizer is developed.

Key words: approximate logic, and-or degree of operator, expert opinion synthesis.

I. Introduction

Neural networks have received widespread attention because of their ability to process inaccurate and fuzzy information. Nevertheless, the use of neural networks to process knowledge is very limited. In order to overcome this weakness, an approximate logic is defined. This logic can properly describe a neural network. It not only has a fuzzy logic value but also a fuzzy logic operator. Finally, a neural-network-based multi-expert opinion synthesis system is also described. It is capable of storing rules and can be trained. The conventional weight method is equivalent to a special case where the neural net does not have any knowledge in storage. After receiving opinions from various experts, it enters a pondering period and then makes the final judgment. This process is similar to that in the human mind and is worth further investigation.

II. Approximate Logic and Neural Network

1. Approximate Logic

An approximate logic system is defined as follows. The logic value of the system is between [0, 1]. By varying some parameters, a function may be switched from "or" to "and," or from "yes" to "no." Different from fuzzy logic, not only its logic value is fuzzy but also its operator can be fuzzy. The following theorems are fairly easy to prove. Due to page limitation, the proof is omitted.

Definition 1. Approximate Logic. The logic value of the system is between [0, 1]. A_1, A_2, ... A_k are the approximate logic variables, tr_i and w_i, i = 1, ..., k, are constants and weights, respectively. The weight is between [-1, 1]:

$$F=\sum_{i=1}^{i=k} w_i * A_i \qquad AF(A_1, A_2, \cdots, A_k)=\begin{cases} F \text{ if } 0\le F\le 1 \text{ and } F\ge tr \\ 1 \text{ if } 1<F \text{ and } F\ge tr \\ 0 \text{ otherwise} \end{cases}$$

It is easy to know that the inverse function AF is a multi-value function.

Theorem 1. Assume that A_i can only be either 0 or 1.

(1) if $tr=1$, $w_i=1$, $i=1, \cdots, k$, then $AF(A_1, \cdots, A_k)=A_1 \vee A_2 \cdots \vee A_k$

(2) if $tr=k$, $w_i=1$, $i=1, \cdots, k$, then $AF(A_1, \cdots, A_k)=A_1 \wedge A_2 \cdots \wedge A_k$

(3) if $w_1=-1$, $w_2=1$, $tr=1$, then $AF(A_1, 1)=\neg A_1$

(4) if $w_1=1$, $tr=0$, then $AF(A_1)=A_1$

Therefore, every Boolean function can be expressed in terms of AF.

Proof: Omitted.

Definition 2. If the weight of a function with n variables belongs to [0, 1] and its threshold is $0 < tr < n$, then the "and/or" operator is defined as follows:

$$\mathrm{aod}(AF) = tr/(\sum_{i=1}^{n} w_i)$$

When aod changes from 0 to 1, the function

$$\vee_{i=1}^{n} A_i \text{ becomes } \wedge_{i=1}^{n} A_i.$$

Through the use of aod, a fuzzy operation can be discretized into corresponding Boolean operations.

2. Neural Network Model

Our neural network is comprised of four parts: a weight set, a nodal point set, a threshold set and an output set. $w_{ij} \in [-1, 1]$ is the weight from node i to j. Let us assume that $w_{ij} \in [0, 1]$. s_i is the excitation value of node i and out_j is the output. Then, the system operates according to the following:

$$s_i(t+1) = \sum_{j=1}^{j=h} w_{ij} * out_j(t) + b_i(t) * ext_i(t) \quad (1)$$

where $b_i(t)$ is a coefficient in [0, 1].

$$out_i(t) = \begin{cases} s_i(t) & \text{if } s_i(t) \geq tr_i \text{ and } \min \leq s_i(t) \leq \max \\ \max & \text{if } s_i(t) > \max \text{ and } s_i(t) \geq tr_i \\ \min & \text{otherwise} \end{cases} \quad (2)$$

In general, max = 1, min = 0, or max = 1, min = -1. The net input of the system is defined as:

$$net_i(t) = \sum_{j \neq i} w_{ij} * out_j(t) + b(t) * ext_i(t) \quad (3)$$

$$s_i(t+1) = net_i(t) + w_{ii} * out_i(t) \quad (4)$$

Since all propositions are self-supporting $w_{ii} \in [0, 1]$.

Definition 3. The Boolean system is a binary system and its output is either 0 or 1. In this case, max = 1, min = 0.

Theorem 2. If a system has max = 1 and min = 0, and each weight belongs to [0, 1], then each node is equivalent to an approximate function.

Proof: Omitted.

Deduction 1. Every Boolean function can be implemented using a Boolean system.

Proof: Omitted.

Theorem 3. If $net_i(t) = \rho$, where ρ is a constant, and $\min < \rho < \max$, then if $\rho + w_{ii} * out_i(t) < tr_i$, then $out_i(t+k) = \min$, $k = 1, 2, \ldots$; if $\rho + w_{ii} * out_i(t) > tr_i$, $w_{ii} = 1$ and $\rho > 0$, then $k > (\max - tr_i)/\rho$ and $out_i(t+k) = \max$. If $w_{ii} = 1$ and $\rho < 0$, then $k > (\max - tr_i)/\rho$ and $out_i(t+k) = \min$. If $\rho + w_{ii} * out_i(t) > tr_i$, and $0 < w_{ii} < 1$ and $\rho/(1 - w_{ii}) > tr_i$, then $out_i(t+k) \rightarrow \rho/(1 - w_{ii})$. When $k \rightarrow +\infty$, if $\rho + w_{ii} * out_i(t) > tr_i$, $0 < w_{ii} < 1$ and $\rho/(1 - w_{ii}) < tr_i$, then $out_i(t+k) \rightarrow \min$ when $k \rightarrow +\infty$.

Proof: Omitted.

Conclusion. If net_i is a constant, then the final out_i is stable.

Definition 4. A neural network can be viewed as a directed graph, called a neural net graph. Each node is a node for the graph. Each non-zero weight is the edge of the graph.

Theorem 4. In a neural network, if all the input ext_i and their weights are constants, and if the neural network graph has no loop, then the system will oscillate in operation.

Proof: Omitted.

Definition 5. The deduction process of the neural network is as follows: A node that does not receive information from other nodes is a conditional node. A conditional node is a premise for the system to accept a deduction. When the system operates for some time, if other nodes have been stabilized, then the state of these nodes is used to derive a conclusion under the premise. A looped neural network is usually a chaotic system and its qualitative stability analysis is very difficult to perform. It usually oscillates because there are often contradictory bits of information.

Definition 6. Linear Stability Condition. If $O = (o_1, o_2, \ldots o_k)$ represents the steady states of the neural network and $\min < o_i < \max$ for all i, then O is known as linear stability.

Theorem 5. Let us assume that in a system the coefficient b(t) of equation (1) is zero, then the condition for linear stability is:

$$\begin{bmatrix} w_{oo} & \cdots & w_{ok} \\ \cdots & \cdots & \cdots \\ w_{ko} & \cdots & w_{kk} \end{bmatrix} * \begin{bmatrix} o_o \\ \vdots \\ o_k \end{bmatrix} = \begin{bmatrix} o_o \\ \vdots \\ o_k \end{bmatrix} \quad (5)$$

Proof: Omitted.

III. Neural-Network-Based Multi-Expert Opinion

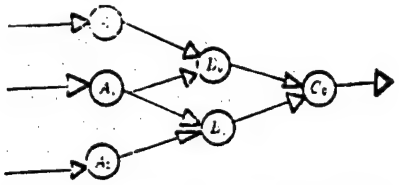


Figure 1. Layered Neural Network

Synthesis System

1. Purpose of Use

Expert systems are being developed from a simple databank of a single expert to a complex system involving multiple experts. In a distributed multi-expert system, the key part is a subsystem that synthesizes the opinions of the experts. Conventional methods, including weighted method, voting method and statistical method, etc. have the following major disadvantages:

(1) There is no mechanism to incorporate experience into the management of synthesis of expert opinions.

(2) It is not possible to provide training.

(3) It is impossible to reflect the situation wherein the experts' opinions cannot be mediated.

Since the ability to synthesize experts' opinions is directly related to human intelligence, it is not possible to create an ideal model for the synthesis of experts' opinions. We believe that if a model can be created, the model ought to have a number of tunable parameters to meet the needs in different disciplines. Furthermore, it ought to be able to overcome the three major disadvantages described above, at least to some extent. This model will be helpful to our effort to improve the capability of the system to synthesize experts' opinions. As a result of our investigation, it was discovered that the competition and coordination mechanism of a neural network can perform these functions well.

2. System Architecture

A state cell is used to represent an expert's judgment. A state cell is connected to a judgment. The value of a state is between [0, 1]. Each cell in the system has an input and ext_i. Various opinions of the experts are sent into the system through these input ends, as shown in Figure 2. In equation (3), b(t) can be considered as the weight for the expert. Prior to operation, common knowledge of the experts is stored in the system. There are two ways to gather experts' opinions. One is a parallel listening method. The method collects all the opinions of the experts and provides the result to the system after applying a weight. This method is very similar to the conventional methods. Its advantage is simplicity. However, its shortcoming is that it cannot reflect disagreement among experts in detail. It is only able to reflect the situation wherein the collective opinion of the experts is consistent with the knowledge stored in the network. The other way is a serial method which listens to experts' opinions one at a time. After listening to the opinion of each expert, the neural network is allowed to complete several cycles of processing before listening to the next expert. The opinions of certain experts can also be listened to repeatedly. This process is very similar to that used by a human. In this serial mode, if $w_{ii} = 1$, $w_{ij} = 0$, i not equal to j, and tr_i = min, the results provided by the system are the weighted values. The parameters of the system must be tuned in practice. The neural network is usually unstable. In other words, it is impossible to inquire about the result after the system has calmed down completely.

In order to solve this problem, a waiting period number T and an observation period number L are specified for the system. After the system listens to the opinion of the last expert, it is allowed to operate for T cycles. This is equivalent to allowing the system to think based on the knowledge it stores. Then, the system is observed over L cycles. During this period, the state of every cell is analyzed. The synthesis result is obtained by finding the mean value of these observed values. It is easy to determine whether a cell is oscillating. As long as the observation period is equal to the oscillation period, a mean value of (max + min)/2 indicates that no judgment can be made.

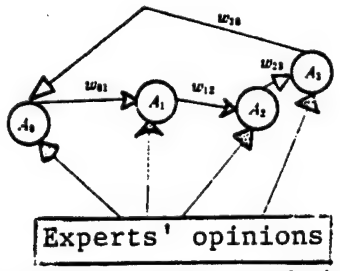


Figure 2. Multi-Expert Opinion Synthesis System

3. Results of Operation

Definition 7. If $S_g = (s_1, s_2, ..., s_k)$ represents the result and $Z_g = (z_1, z_2, ..., z_k)$ is the experts' opinion, the degree of acceptance of experts' opinion is defined as follows:

$$err = \sum_{i=1}^{i=k} (z_i - s_i)^2$$

Based on simulation, the following interesting phenomena have been discovered:

(1) Self-contradictory opinions produce a no decision or oscillation.

If contradictory knowledge A $\rightarrow$, ..., $\rightarrow$ > A is stored into the system, the system oscillates.

(2) Winning with masses.

When a number of experts participate in a debate, if the opinion of the majority is consistent, the output of the system is in agreement with the majority opinion. Of course, the majority opinion must be consistent with the knowledge learned by the system. In Example 1, $w_{01} = 0.9$, $w_{12} = -0.8$, $w_{32} = 0.8$ and $w_{31} = 0.9$. Other weights are 0 and all thresholds are -3.

From Tables 1 and 2, we can see that the opinion of expert Z_0 is different from that of others. The final operating result is also different.

(3) Winning with few.

Since the system "ponders" for several cycles after listening to the opinions of all experts, during this period, the system forgets the opinions of certain experts that are incompatible with the knowledge stored in it. Therefore, the system sometimes reaches the conclusion that is consistent with the minority, as in Example 2.

In this system, $w_{01} = 0.9$, $w_{12} = -0.8$, $w_{32} = 0.8$ and $w_{31} = 0.9$. Other weights are 0 and all thresholds are -3.

From Tables 3 and 4 we can see that the viewpoint of expert Z_2 is different from that of others, but is consistent with the knowledge stored in the system. The final result is in agreement with his opinion.

Table 1. Results of Example 1

cycle	0	8	11	14	22	24	27	30	32
expert	Z_0	Z_0	Z_1	Z_1	Z_1	Z_1	Z_1	Z_1	Z_1
b(t)	0.60	0.60	0.50	0.50	0.40	0.40	0.30	0.30	0.30
node1	0.00	-0.39	0.40	0.40	0.28	0.28	0.21	0.21	0.21
node2	0.00	-0.22	1.00	1.00	0.74	0.74	0.58	0.58	0.58
node3	0.00	0.32	-0.82	-0.82	-0.64	-0.64	-0.40	-0.40	-0.40
node4	0.00	-0.39	0.45	0.45	0.32	0.32	0.27	0.27	0.27

Table 2. Opinions of Experts in Example 1

node	A_0	A_1	A_2	A_3	err
Z0	-0.65	0.80	0.75	-0.65	2.95
Z1	0.80	0.65	-0.75	0.90	0.85
Z2	0.70	0.50	-0.75	0.80	0.65
Z3	0.70	0.50	-0.49	0.90	0.65

Table 3. Results of Example 2

cycle	0	8	11	18	23	26	29	31
expert	Z_0	Z_0	Z_1	Z_1	Z_2	Z_2	Z_2	Z_2
b(t)	0.60	0.60	0.50	0.40	0.40	0.30	0.30	0.30
node0	0.00	0.36	0.47	0.34	0.34	0.30	0.30	0.30
node1	0.00	0.23	0.59	0.93	0.93	0.37	0.37	0.37
node2	0.00	-0.07	0.14	-0.58	-0.58	-0.22	-0.22	-0.22
node3	0.00	0.36	0.46	0.38	0.38	0.28	0.28	0.28

Table 4. Opinions of Experts in Example 2

node	A_0	A_1	A_2	A_3	err
Z_0	0.60	-0.70	-0.30	0.60	1.35
Z_1	0.95	-0.50	0.50	0.91	2.10
Z_2	0.85	0.70	-0.35	0.95	0.87
Z_3	0.99	-0.50	-0.50	0.95	1.76

4. Learning From Samples

As described earlier, the system architecture includes the interconnection among various determinations. These interconnections need to be corrected through a learning process in operation. The learning process was completed by adopting the Hebb or Delt rule. If the opinion of a certain expert is extremely important, it is repeatedly used to stimulate the system until the system accepts his opinion.

Assuming $X = (x_1, x_2, \ldots x_k)$ is the opinion of that expert, then the weight from the ith to the jth cell becomes:

Hebb rule:

$$\text{middle} = (\max + \min)/2;$$

$$\Delta W_{ij}(t+1) = \mu * (x_i - \text{middle}) * (x_j - \text{middle});$$

$$W_{ij}(t+1) = \alpha * w_{ij}(t) + \Delta W_{ij}(t+1)$$

α and μ belong [0, 1].

Delt rule:

$$out_i = AF_i(x_1, x_2, \cdots, x_k);$$

$$\Delta W_{ij}(t+1) = \mu * (x_i - out_i) * out_j;$$

$$W_{ij}(t+1) = \alpha * w_{ij}(t) + \Delta W_{ij}(t+1)$$

α and μ belong to [0, 1]. Through the use of an auxiliary node R_n, threshold learning can be changed to weight learning. In this work, the threshold tr_i = middle and $out_{Rn} = 1$. When it is necessary to learn the threshold, the following rule may apply: Assuming AF_i is the approximate logic function of node i, then

if $AF_i > \min$ and $x_i = \min$, then $tr_{new\ i}$ [i is subscript for new] = $AF_i(x_1, x_2, \ldots, x_k) + \mu * \max$;

if $AF_i = \min$ and $x_i > \min$, then $tr_{new\ i}$ [i is subscript for new] = x_i.

References

1. D. E. Rumelhart, J. L. McClelland, PDP Research Group, "Parallel Distributed Processing," Volume 1: Foundations.

2. Lu Ru Qian, "Distributed Knowledge-Base System," in COMPUTER RESEARCH AND DEVELOPMENT, Supplement, 1988.

3. Shi Zhongzhi, "Neural Computing," Graduate School, University of Science and Technology of China, 1991.

Hu Hong graduated from the Department of Computer Science at Zhejiang University in 1984. From 1984 to 1987, he was involved in computer hardware design at the CAS Institute of Computing Technology. Since 1987, he has been working on artificial intelligence.

Shi Zhongzhi was born in Jiaxing, Jiangsu in 1941. He is a Research Fellow at the National Research Center for Intelligent Computing Systems, CAS Institute of Computing Technology, part-time professor at the graduate school of USTC and Dalian Institute of Technology, chairman of IFIP automatic inference group, vice chairman of the China Artificial Intelligence Society, editor of journals such as MOSHI SHIBIE YU RENGONG ZHINENG [PATTERN RECOGNITION AND ARTIFICIAL INTELLIGENCE]. He has been studying artificial intelligence, neural networks and databases for long periods of time. He has authored six books and published over 150 papers.

General-Purpose Parallel Neural Network Simulation System $GP^2N^2S^2$

Main Details

93P60120A Shenyang XIAOXING WEIXING JISUANJI XITONG [MINI-MICRO SYSTEMS] in Chinese Vol 13 No 12, Dec 92 pp 16-21, 32

[Article by Chen Guoliang [7115 0948 5328], Xiong Yan [3574 3543], and Fang Xiang [2455 4382] of the Dept. of Computer Science, University of Science & Technology of China (USTC), Hefei 230027: "General-Purpose Parallel Neural Network Simulation System $GP^2N^2S^2$," supported by grants from the State S&T Commission's Basic Research and High Technology Department and the 863 Plan; MS received 21 Jun 92. Cf. brief early report in JPRS-CST-92-010, 22 May 92 p 27]

[Abstract] A Transputer-array-based General-Purpose Parallel Neural Network Simulation System ($GP^2N^2S^2$) developed at USTC is described. This system provides the neural network simulation language, editor, compiler, and user's executive program environment. The user's program, written in sequential code, can be automatically implemented in parallel on a parallel Transputer system. $GP^2N^2S^2$ can simulate not only several currently popular neural networks, but also new neural network models independently developed by the user.

The $GP^2N^2S^2$ hardware configuration and system software module schematic are shown below in Figures 1 and 3, respectively. In Figure 3, PEM is the program editing module, PCM is the program compiling module, CSM is the control simulation module, GDM is the graphics display module, ARCM is the array reconfiguration module, ALM is the array loading module, SCM is the simulation command module, NAM is the neuron allocation module, SIDSM is the sampling input and data sending module, SRM is the simulation results module, PCRTCM is the personal computer host to root Transputer communications module, NC is the network communications subprocess, SC is the simulation computation subprocess, and DA is the data access subprocess. Figure 2, not reproduced, shows the system software structure. Figures 4 and 5, reproduced below, show the usual Transputer array configuration and the $GP^2N^2S^2$ Transputer array configuration, respectively.

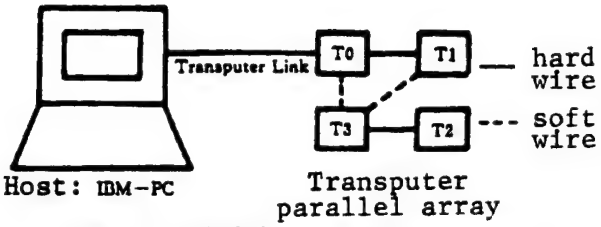


Figure 1. $GP^2N^2S^2$ Hardware Configuration

In the usual configuration, the IBM PC 386 host is connected to a T800 host Transputer, which is connected in turn to other T800 Transputers including the root Transputer; in the $GP^2N^2S^2$ configuration, however, the host is connected directly to the T800 root Transputer, which in turn is connected to the three other T800 Transputers in the four-Transputer array.

$GP^2N^2S^2$ comes with a Neural Network Description Language (NNDL) and Occam 2, can simulate 12,000 neurons, permits over 300,000 connection weights, and has a processing speed of 80,000 IPS (interconnection weight updates per second); it can simulate a 10-city TSP (Traveling Salesman Problem) in 12 seconds.

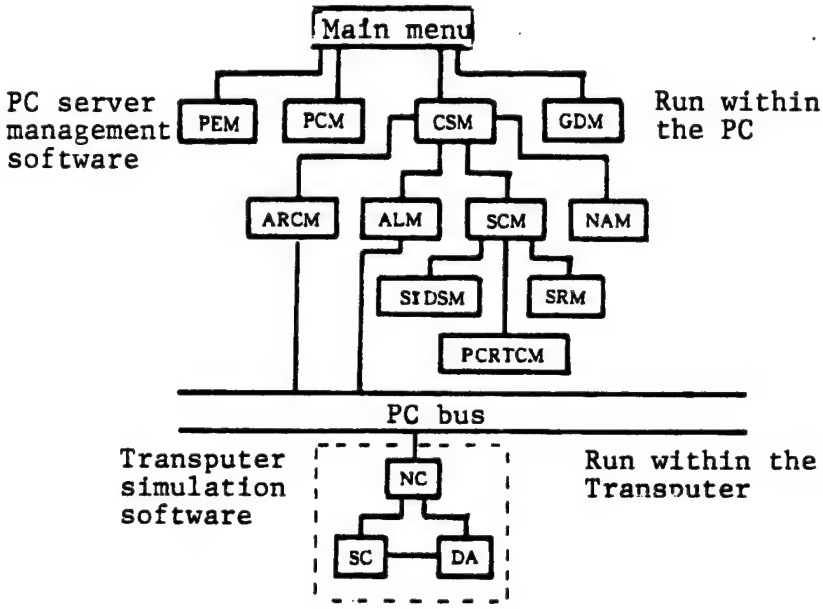


Figure 3. System Software Module Schematic

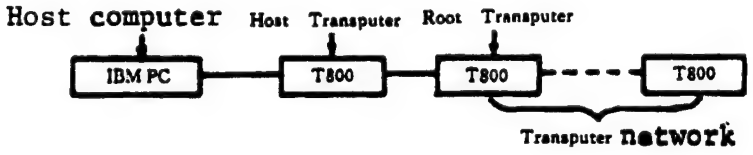


Figure 4. Usual Transputer Array Configuration

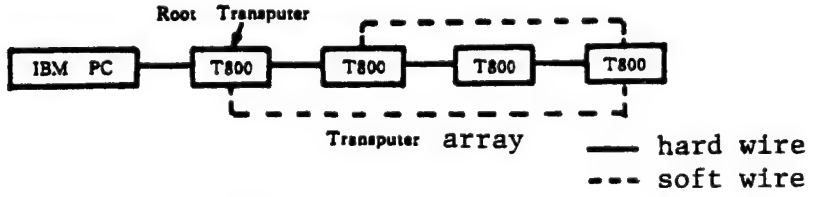


Figure 5. $GP^2N^2S^2$ Transputer Array Configuration

References

1. C. Lynne D'Autrechy, James A. Reggia, Granger G. Sutton II and Sharon M. Goodall, "A General-Purpose Simulation Environment for Developing Connectionist Models," SIMULATION 51:1, 1988.6, pp 5-19.

2. Andrew B. Smith, "A Parallel PDP Network Simulator," IEEE First Intl. Conf. on Neural Networks, Vol II, 1987, pp 377-384.

3. Simon C. J. Garth, "A Chipset for High-Speed Simulation of Neural Network System," IEEE First Intl. Conf. on Neural Networks, Vol II, 1987, pp 443-452.

4. Claude A. Cruz, William A. Hanson, Jason Y. Tam, "Neural Network Emulation Hardware Design Considerations," IEEE First Intl. Conf. on Neural Networks, Vol II, 1987, pp 427-434.

5. William A. Hanson, Claude A. Cruz, Jason Y. Tam, "CONE—Computational Network Environment," IEEE First Intl. Conf. on Neural Networks, Vol II, 1987, pp 531-538.

6. H. P. Siemon, A. Ultsch, "Kohonen Network on Transputers: Implementation and Animation," IEEE Intl. Conf. on Neural Networks, 1990, pp 643-646.

7. "RHINE—Recognition Heuristics Inference and Neuro Computing Software Development Environment," COMPUTROL, No 29, 1990. (Japan, Research Center)

8. Chen Guoliang, "Neural Networks and Their Learning Algorithms," Neural Networks and Their Applications, USTC Publishing House, 1992.2, pp 121-157.

9. Yao Xin, Chen Guoliang, "Neurocomputers," ibid., pp 186-205.

10. IMSBOO8 User's Guide and Reference Manual, INMOS, 1988.

11. INMOS Ltd., Transputer Development System, Second Edition, 1988.

12. Transputer Hardware Handbook, MMEI Institute 14.

13. "General-Purpose Parallel Neural Network Simulation System," Technical Report, USTC Computer Dept., 1992.2.

14. "General-Purpose Parallel Neural Network Simulation System," User's Handbook, USTC Computer Dept., 1992.3.

15. "General-Purpose Parallel Neural Network Simulation System," Programmer's Editing Guide, USTC Computer Dept., 1992.3.

16. "General-Purpose Parallel Neural Network Simulation System," High-Level Neural Network Language, USTC Computer Dept., 1992.3.

Advanced NN Description Language, Editor/Compiler

93P60120B Shenyang XIAOXING WEIXING JISUANJI XITONG [MINI-MICRO SYSTEMS] in Chinese Vol 13 No 12, Dec 92 pp 22-28

[Article by Fan Haibo [5400 3189 3134] and Chen Guoliang of the Dept. of Computer Science, USTC, Hefei 230027: "Advanced Neural Network Description Language, Its Editor/Compiler," supported by grants from the State S&T Commission's Basic Research and High Technology Dept. and the 863 Plan; MS received 21 Jun 92]

[Abstract] Advanced (i.e., high-level) Neural Network Description Language (NNDL) is a non-procedural language—one therefore quite different from traditional (procedural) program design languages such as Fortran and C—provided by GP²N²S² and specially designed for writing neural network simulation programs at the network, layer, and node levels. Several distinctive features of NNDL and its editor/compiler are described.

Although the article has no numbered figures or tables as such, a programming example is provided in which the BP (backpropagation) algorithm is used to solve a mirror-image symmetry problem. This problem is described as follows: if the input binary sequence is center-symmetric, then the output is a 1, otherwise it is 0. For instance, if the input is 011110, the output is 1, but if the input is 011000, the output is 0. Using 6-bit binary sequences, the mirror-image symmetry problem is coded with the following program:

In the diagram, the first six nodes in the input layer are the input binary sequence, while the final two nodes are the virtual nodes, representing the input-layer threshold and intermediate-layer threshold, respectively; both are fixed as an input of -1.

References

1. C. D'Autrechy, J. Reggia, G. Sutton II, S. Goodall, "A General-Purpose Simulation Environment for Developing Connectionist Models," Dept. of Computer Science, Univ. of Maryland, 1988.

2. S. Small, L. Shastri, M. Brucks, S. Kaufman, G. Cottrel, "ISCON—A Network Construction Aid and Simulator for Connectionist Models," Dept. of Computer Science, Univ. of Rochester, 1988.

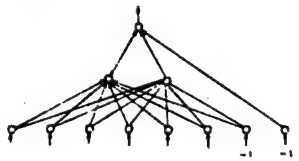
3. University of Science & Technology for National Defense (USTND), Computer Dept., "A Large General-Purpose Neural Network Simulation System," 1990.

4. USTC Computer Dept., "GP²N²S² High-Level Neural Network Description Language," 1992.2.

5. Chen Yiyun, Ma Wanli, "Compiling Principles and Techniques," USTC Publishing House, 1989.

```
network BP
        has layers 3;
        contains layer [1: 3];
        network_model bp;
        nonlinearity sigmoid 1 (1. 0);
        float paras (0. 3);

layer layer1 {
        has nodes 6;
        contains model [1: 6];
        connects layer2 : many_to_many : random (-. 3 ; . 3),
                  layer3 : many_to_many : random (-. 3 ; . 3);
        attribute input;
}

layer layer2 {
        has nodes 2;
        contains node2 [1: 2];
        connects layer3 : all : random (-. 3 ; . 3);
        attribute implicit;
}

layer layer3 {
        has nodes 1;
        contains node31;
        attribute output;
}

nodes model [1: 7] {
        member layer1;
        all connect layer2 (node2 [1: 2]);
}

node node118 {
        member layer1;
        connects layer3 (node31);
}

commands {
    learn {
        sweep_interval 10.          sweep_total 2000;
        samples_expectations : 64
          ( (0. 0. 0. 0. 0. 0. -1. -1)   (1)),    ( (0. 0. 0. 0. 0. 1. -1. -1)   (0)),
          ( (0. 0. 0. 0. 1. 0. -1. -1)   (0)),    ( (0. 0. 0. 0. 1. 1. -1. -1)   (0)),
          ( (0. 0. 0. 1. 0. 0. -1. -1)   (0)),    ( (0. 0. 0. 1. 0. 1. -1. -1)   (0)),
          ( (0. 0. 0. 1. 1. 0. -1. -1)   (0)),    ( (0. 0. 0. 1. 1. 1. -1. -1)   (0)),
          ( (0. 0. 1. 0. 0. 0. -1. -1)   (0)),    ( (0. 0. 1. 0. 0. 1. -1. -1)   (0)),
          ( (0. 0. 1. 0. 1. 0. -1. -1)   (0)),    ( (0. 0. 1. 0. 1. 1. -1. -1)   (0)),
          ( (0. 0. 1. 1. 0. 0. -1. -1)   (1)),    ( (0. 0. 1. 1. 0. 1. -1. -1)   (0)),
          ( (0. 0. 1. 1. 1. 0. -1. -1)   (0)),    ( (0. 0. 1. 1. 1. 1. -1. -1)   (0)),
          ( (0. 1. 0. 0. 0. 0. -1. -1)   (0)),    ( (0. 1. 0. 0. 0. 1. -1. -1)   (0)),
          ( (0. 1. 0. 0. 1. 0. -1. -1)   (0)),    ( (0. 1. 0. 0. 1. 1. -1. -1)   (0)),
          ( (0. 1. 0. 1. 0. 0. -1. -1)   (0)),    ( (0. 1. 0. 1. 0. 1. -1. -1)   (0)),
          ( (0. 1. 0. 1. 1. 0. -1. -1)   (0)),    ( (0. 1. 0. 1. 1. 1. -1. -1)   (0)),
          ( (0. 1. 1. 0. 0. 0. -1. -1)   (0)),    ( (0. 1. 1. 0. 0. 1. -1. -1)   (0)),
          ( (0. 1. 1. 0. 1. 0. -1. -1)   (0)),    ( (0. 1. 1. 0. 1. 1. -1. -1)   (0)),
          ( (0. 1. 1. 1. 0. 0. -1. -1)   (0)),    ( (0. 1. 1. 1. 0. 1. -1. -1)   (0)),
          ( (0. 1. 1. 1. 1. 0. -1. -1)   (0)),    ( (0. 1. 1. 1. 1. 1. -1. -1)   (1)),
          ( (1. 0. 0. 0. 0. 0. -1. -1)   (0)),    ( (1. 0. 0. 0. 0. 1. -1. -1)   (0)),
          ( (1. 0. 0. 0. 1. 0. -1. -1)   (0)),    ( (1. 0. 0. 0. 1. 1. -1. -1)   (0)),
          ( (1. 0. 0. 1. 0. 0. -1. -1)   (0)),    ( (1. 0. 0. 1. 0. 1. -1. -1)   (0)),
          ( (1. 0. 0. 1. 1. 0. -1. -1)   (0)),    ( (1. 0. 0. 1. 1. 1. -1. -1)   (0)),
          ( (1. 0. 1. 0. 0. 0. -1. -1)   (0)),    ( (1. 0. 1. 0. 0. 1. -1. -1)   (0)),
          ( (1. 0. 1. 0. 1. 0. -1. -1)   (0)),    ( (1. 0. 1. 0. 1. 1. -1. -1)   (0)),
          ( (1. 0. 1. 1. 0. 0. -1. -1)   (0)),    ( (1. 0. 1. 1. 0. 1. -1. -1)   (0)),
          ( (1. 0. 1. 1. 1. 0. -1. -1)   (0)),    ( (1. 0. 1. 1. 1. 1. -1. -1)   (0)),
          ( (1. 1. 0. 0. 0. 0. -1. -1)   (0)),    ( (1. 1. 0. 0. 0. 1. -1. -1)   (0)),
          ( (1. 1. 0. 0. 1. 0. -1. -1)   (0)),    ( (1. 1. 0. 0. 1. 1. -1. -1)   (0)),
          ( (1. 1. 0. 1. 0. 0. -1. -1)   (0)),    ( (1. 1. 0. 1. 0. 1. -1. -1)   (0)),
          ( (1. 1. 0. 1. 1. 0. -1. -1)   (0)),    ( (1. 1. 0. 1. 1. 1. -1. -1)   (0)),
          ( (1. 1. 1. 0. 0. 0. -1. -1)   (0)),    ( (1. 1. 1. 0. 0. 1. -1. -1)   (0)),
          ( (1. 1. 1. 0. 1. 0. -1. -1)   (0)),    ( (1. 1. 1. 0. 1. 1. -1. -1)   (0)),
          ( (1. 1. 1. 1. 0. 0. -1. -1)   (0)),    ( (1. 1. 1. 1. 0. 1. -1. -1)   (0)),
          ( (1. 1. 1. 1. 1. 0. -1. -1)   (0)),    ( (1. 1. 1. 1. 1. 1. -1. -1)   (1));
    }
    spread {
        samples : 64
                (0. 0. 0. 0. 0. 0. -1. -1),    ( (0. 0. 0. 0. 0. 1. -1. -1),
                (0. 0. 0. 0. 1. 0. -1. -1),    ( (0. 0. 0. 0. 1. 1. -1. -1),
                (0. 0. 0. 1. 0. 0. -1. -1),    ( (0. 0. 0. 1. 0. 1. -1. -1),
                (0. 0. 0. 1. 1. 0. -1. -1),    ( (0. 0. 0. 1. 1. 1. -1. -1),
                (0. 0. 1. 0. 0. 0. -1. -1),    ( (0. 0. 1. 0. 0. 1. -1. -1),
                (0. 0. 1. 0. 1. 0. -1. -1),    ( (0. 0. 1. 0. 1. 1. -1. -1),
                (0. 0. 1. 1. 0. 0. -1. -1),    ( (0. 0. 1. 1. 0. 1. -1. -1),
                (0. 0. 1. 1. 1. 0. -1. -1),    ( (0. 0. 1. 1. 1. 1. -1. -1),
                (0. 1. 0. 0. 0. 0. -1. -1),    ( (0. 1. 0. 0. 0. 1. -1. -1),
                (0. 1. 0. 0. 1. 0. -1. -1),    ( (0. 1. 0. 0. 1. 1. -1. -1),
                (0. 1. 0. 1. 0. 0. -1. -1),    ( (0. 1. 0. 1. 0. 1. -1. -1),
                (0. 1. 0. 1. 1. 0. -1. -1),    ( (0. 1. 0. 1. 1. 1. -1. -1),
                (0. 1. 1. 0. 0. 0. -1. -1),    ( (0. 1. 1. 0. 0. 1. -1. -1),
                (0. 1. 1. 0. 1. 0. -1. -1),    ( (0. 1. 1. 0. 1. 1. -1. -1),
                (0. 1. 1. 1. 0. 0. -1. -1),    ( (0. 1. 1. 1. 0. 1. -1. -1),
                (0. 1. 1. 1. 1. 0. -1. -1),    ( (0. 1. 1. 1. 1. 1. -1. -1),
                (1. 0. 0. 0. 0. 0. -1. -1),    ( (1. 0. 0. 0. 0. 1. -1. -1),
                (1. 0. 0. 0. 1. 0. -1. -1),    ( (1. 0. 0. 0. 1. 1. -1. -1),

                (1. 0. 0. 1. 0. 0. -1. -1),    ( (1. 0. 0. 1. 0. 1. -1. -1),
                (1. 0. 0. 1. 1. 0. -1. -1),    ( (1. 0. 0. 1. 1. 1. -1. -1),
                (1. 0. 1. 0. 0. 0. -1. -1),    ( (1. 0. 1. 0. 0. 1. -1. -1),
                (1. 0. 1. 0. 1. 0. -1. -1),    ( (1. 0. 1. 0. 1. 1. -1. -1),
                (1. 0. 1. 1. 0. 0. -1. -1),    ( (1. 0. 1. 1. 0. 1. -1. -1),
                (1. 0. 1. 1. 1. 0. -1. -1),    ( (1. 0. 1. 1. 1. 1. -1. -1),
                (1. 1. 0. 0. 0. 0. -1. -1),    ( (1. 1. 0. 0. 0. 1. -1. -1),
                (1. 1. 0. 0. 1. 0. -1. -1),    ( (1. 1. 0. 0. 1. 1. -1. -1),
                (1. 1. 0. 1. 0. 0. -1. -1),    ( (1. 1. 0. 1. 0. 1. -1. -1),
                (1. 1. 0. 1. 1. 0. -1. -1),    ( (1. 1. 0. 1. 1. 1. -1. -1),
                (1. 1. 1. 0. 0. 0. -1. -1),    ( (1. 1. 1. 0. 0. 1. -1. -1),
                (1. 1. 1. 0. 1. 0. -1. -1),    ( (1. 1. 1. 0. 1. 1. -1. -1),
                (1. 1. 1. 1. 0. 0. -1. -1),    ( (1. 1. 1. 1. 0. 1. -1. -1),
                (1. 1. 1. 1. 1. 0. -1. -1),    ( (1. 1. 1. 1. 1. 1. -1. -1);
            }
}
```

The network topological structure is shown in the following diagram:



Parallel Simulation Controller

93P60120C Shenyang XIAOXING WEIXING JISUANJI XITONG [MINI-MICRO SYSTEMS] in Chinese Vol 13 No 12, Dec 92 pp 29-32

[Article by Huang Junbin [7806 0193 2430] and Chen Guoliang of the Dept. of Computer Science, USTC, Hefei 230027: "Design, Implementation of $GP^2N^2S^2$ Parallel Simulation Controller," supported by grants from the State S&T Commission's Basic Research and High Technology Dept. and the 863 Plan; MS received 21 Jun 92]

[Abstract] $GP^2N^2S^2$'s core software—the parallel simulation controller (PSC)—is introduced, and its design concept and implementation are described. The PSC includes three parts: the NC (sub)process, the DA (sub-)process, and the SC (sub)process, all three executed in parallel within the Transputer array.

One figure (not reproduced) shows the Transputer array.

References

1. James L. McClelland and David E. Rumelhart, "Explorations in Parallel Distributed Processing," 1988.

2. $GP^2N^2S^2$ Overall Design Report, USTC Computer Dept., 1992.2.

3. "Large General-Purpose Neural Network Simulation System," USTND, 1991.4.

4. $GP^2N^2S^2$ Programmer's Editing Guide, USTC Computer Dept., 1992.3.

Central Control Block

93P60120D Shenyang XIAOXING WEIXING JISUANJI XITONG [MINI-MICRO SYSTEMS] in Chinese Vol 13 No 12, Dec 92 pp 33-36

[Article by Zhang Qingjun [1728 1987 6511] and Chen Guoliang of the Dept. of Computer Science, USTC, Hefei 230027: "Design, Implementation of $GP^2N^2S^2$ Central Control Block," supported by grants from the State S&T Commission's Basic Research and High Technology Dept. and the 863 Plan; MS received 21 Jun 92]

[Abstract] The Central Control Block (CCB) is the center of both the data stream and control stream of $GP^2N^2S^2$, which is a MIMD [multiple instruction stream/multiple data stream] system. The CCB handles system data collection, processing, sending, tasks assignment, etc. The implementation of the CCB is detailed, its task assignment strategy is discussed, and the system datagram protocol related to the CCB implementation is described.

The $GP^2N^2S^2$ system software consists of five large modules: the compiler module (NNC), the CCB, the PSC, the algorithm library (LIB), and the dynamic graphics simulator (part of the System Integrated Environment, or SIE), as shown in Figure 1 below. The CCB itself consists of three parts, labeled CCB1, CCB2, and CCB3 in the figure. Table 1 (not reproduced) lists NNC output data and corresponding NIP (Network Information Package) data and NSP (Network Structure Package) data, while Table 2 (not reproduced) lists NTP (Network State Package) and DIP (Dynamic Information Package) data.

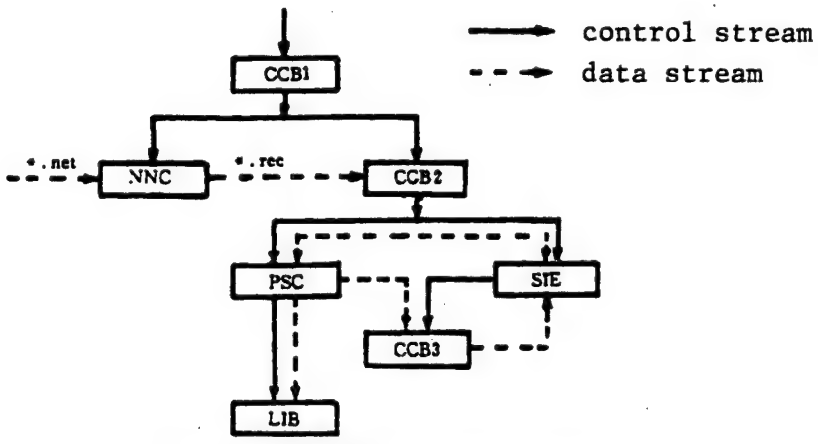


Figure 1. $GP^2N^2S^2$ System Software Structure

References

1. $GP^2N^2S^2$ Development Report, USTC Computer Dept., 1992.1.

2. $GP^2N^2S^2$ Technical Report, USTC Computer Dept., 1992.3.

3. $GP^2N^2S^2$ User's Manual, USTC Computer Dept., 1992.3.

Algorithmic Library, Applications

93P60120E Shenyang XIAOXING WEIXING JISUANJI XITONG [MINI-MICRO SYSTEMS] in Chinese Vol 13 No 12, Dec 92 pp 37-43, 48

[Article by Zhang Qun [1728 5028] and Chen Guoliang of the Dept. of Computer Science, USTC, Hefei 230027: "Establishment of the $GP^2N^2S^2$ Algorithmic Library, Its Applications," supported by grants from State S&T Commission's Basic Research and High Technology Dept. and the 863 Plan; MS received 1 Jun 92]

[Abstract] The $GP^2N^2S^2$ algorithm library (LIB) includes several popular neural network algorithms, such as BP (Backpropagation), Kohonen, and Hopfield. These have been employed to solve some real problems such as the TSP, Associative Memory (AM) and mirror symmetry. Also, the dynamic code loading method has been used to implement simulation calling for user-defined algorithms.

Ten figures (not reproduced) show a BP network, a single-node calculation flow chart for a BP algorithm, a mirror-symmetry network structure, Rumelhart's simulation results, USTC's simulation results, a Kohonen network, a single-node calculation flow chart for the Kohonen algorithm, a Kohonen network used to solve the TSP, standard samples of numerals for NN recognition, and the simulation results for noisy numeral input samples.

References

1. Chen Guoliang, "Neural Networks and Their Learning Algorithms," Neural Networks and Their Applications, USTC Publishing House, 1992, pp 121-157.

2. Richard P. Lippmann, "An Introduction to Computing With Neural Nets," IEEE ASSP MAGAZINE, Vol 3, No 4, pp 4-22, April 1987.

3. Wang Yang, Guang Wei, Zhuang Lingzhu, "Parallel Processor Transputers and the Occam Language," Marine Simulator Joint Company, Sep. 1988.

4. David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning Representations by Back-Propagating Errors," NATURE, Vol 323, 9 Oct. 1986.

5. Jiao Licheng, "Theory of Neural Network Systems," Xidian University Publishing House, Dec. 1990.

Integrated Environment, Dynamic Graphical Simulator

93P60120F Shenyang XIAOXING WEIXING JISUANJI XITONG [MINI-MICRO SYSTEMS] in Chinese Vol 13 No 12, Dec 92 pp 44-48

[Article by Qin Xiaoou [4440 1420 7743] and Chen Guoliang of the Dept. of Computer Science, USTC, Hefei 230027: "Design, Characteristics of $GP^2N^2S^2$ Integrated Environment, Dynamic Graphical Simulator," supported by grants from the State S&T Commission's Basic Research and High Technology Dept. and the 863 Plan; MS received 1 Jun 92]

[Abstract] The $GP^2N^2S^2$ system integrated environment (SIE) provides users with a common interface supporting editing, compiling, and running. The dynamic graphical simulator which realizes the dynamic procedure simulation of artificial NNs is driven by and belongs to this environment. The design and features of the two parts are discussed.

Two figures, both reproduced below, show the architecture of the SIE and the structure of the dynamic simulation controller.

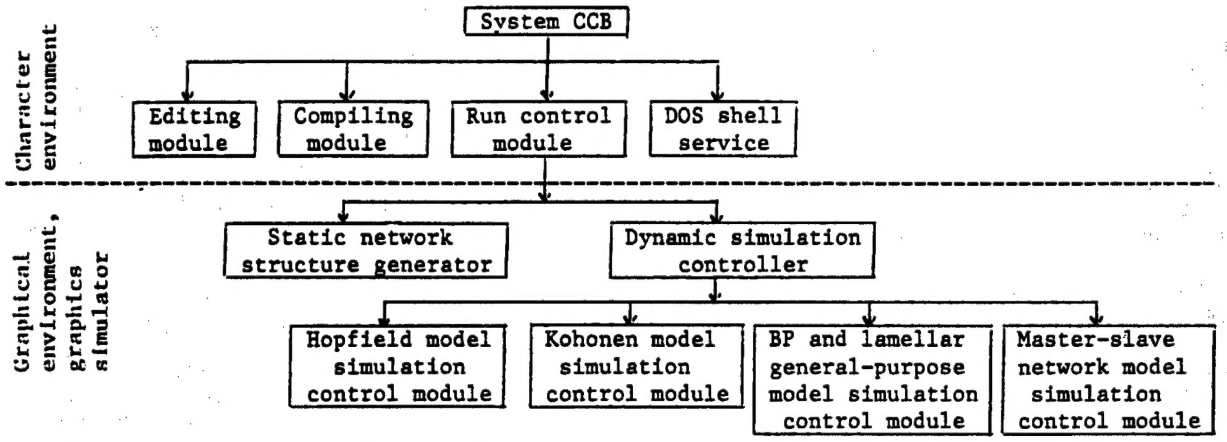


Figure 1. Architecture of SIE

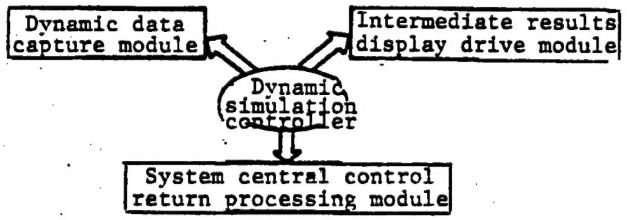


Figure 2. Structure of Dynamic Simulation Controller

References

1. Pan Jingui et al., "Turbo C Program Design Techniques," Nanjing University Publishing House, 1989.

2. Xie Junyuan et al., "Turbo Prolog Program Design Techniques," Nanjing University Publishing House, 1989.

3. Borland International, Inc., Turbo C User's Guide, 4585 Scotts Valley Drive, Scott Valley, CA 95066, USA, 1987.

On Design, Analysis of MPLPC [Multi-Pulse-Excitation Linear Predictive Coding] Vector Quantizer Based on Artificial Neural Network

40100052A Beijing TONGXIN XUEBAO [JOURNAL OF CHINA INSTITUTE OF COMMUNICATIONS] in Chinese Vol 13 No 5, Sep 92 pp 3-10

[English abstract of article by Xu Bingzheng and Peng Lei of South China University of Technology, Guangzhou; MS received 18 Oct 91]

[Text] The application of neural networks in speech compression encoding is discussed. A vector quantizer based on an artificial neural network is provided which can be used in the quantization of multi-pulse excitation analysis model of the speech signal. The quantizing network is somewhat similar to Kohonen's net. It performs the parameter analysis and quantizing process together. Compared to the traditional method which performs analysis first and then quantization, it has some excellent properties. We provide the architecture and learning rule of the quantizing network, and compare it with the traditional method in the implementation. Finally, we simulate the quantizing network with practical speech signal. The experimental results show that our scheme is feasible.

Modified Kohonen Self-Organizing Neural Network, Adaptive Vector Quantization of Images

40100052B Beijing TONGXIN XUEBAO [JOURNAL OF CHINA INSTITUTE OF COMMUNICATIONS] in Chinese Vol 13 No 5, Sep 92 pp 16-21

[English abstract of article by Wang Wei, Cai Dejun, and Wan Faguan of Huazhong University of Science and Technology, Wuhan; MS received 14 Oct 91]

[Text] Based on discussion of the principle of Kohonen's self-organizing feature maps (SOFM), a modified SOFM (MSOFM) algorithm is proposed to reduce blocking effect of vector quantization of images. Two eigenvalues are designed in DCT (Discrete Cosine Transform)

domain to classify image blocks, then the application of MSOFM algorithm in adaptive vector quantization is discussed. The results of computer simulation show that the MSOFM training algorithm significantly reduces blocking effect and has better performance than the SOFM algorithm.

New Models of Holographic Network, Hamming Net, Their Application in Handwritten Chinese-Character Recognition

40100052C Beijing TONGXIN XUEBAO [JOURNAL OF CHINA INSTITUTE OF COMMUNICATIONS] in Chinese Vol 13 No 5, Sep 92 pp 54-59

[English abstract of article by Yu Yinglin and Deng Da of South China University of Technology, Guangzhou; MS received 16 Mar 92]

[Text] We propose two new methods, one for holographic memory, another of a fast-converging Hamming net, both with an automatic attention moving function. Satisfactory results have been obtained after the construction of a handwritten Chinese-character recognition system using these new models.

Applications of Neural Network for Handwritten Digit Recognition

40100052D Beijing TONGXIN XUEBAO [JOURNAL OF CHINA INSTITUTE OF COMMUNICATIONS] in Chinese Vol 13 No 5, Sep 92 pp 60-64

[English abstract of article by Wang Minghui, Pan Xinan, and Shen Min of Beijing University of Posts and Telecommunications; MS received 2 Mar 92]

[Text] A feedforward multilayer neural network with backpropagation learning algorithm to recognize handwritten digits written by 40 persons is used. First, a HP scanner converts the original digit images to binary images. Then, some additional preprocessing is performed to segment and normalize the digits, the binary images are scaled to a 32 x 32 pixel matrix, and the features are extracted to change the representation of a digit from a pixel matrix to a feature description. Finally, a result of 0.4 percent error rate at 25 percent reject rate in the computer simulation is achieved. Some problems encountered in the experiment are also discussed.

Learning Algorithm for Speech Recognition With Recurrent Neural Network

40100052E Beijing TONGXIN XUEBAO [JOURNAL OF CHINA INSTITUTE OF COMMUNICATIONS] in Chinese Vol 13 No 5, Sep 92 pp 76-79

[English abstract of article by Li Haizhou and Xu Bingzheng of South China University of Technology, Guangzhou; MS received 2 Nov 91]

[Text] Learning to associate static input/output pairs can be accomplished with layered connectionist networks with feedforward links alone, but feedback links are required to provide the network with state sequence information, in order to capture sequential behavior. In this paper, a multilayer network architecture with dynamic neurons which have multiple local feedbacks is built. The network proposed can be trained to memorize sequential patterns. A new learning algorithm is also derived which is more effective and easier to implement. Finally, some experiments on speech recognition of Chinese numbers are designed to explore the capabilities of proposed networks to learn dynamic properties of time-varying data. The performance of dynamic neurons with different time delay periods is also shown.

Foreign Broadcast Information Service (FBIS) and Joint Publications Research Service (JPRS) publications contain political, military, economic, environmental, and sociological news, commentary, and other information, as well as scientific and technical data and reports. All information has been obtained from foreign radio and television broadcasts, news agency transmissions, newspapers, books, and periodicals. Items generally are processed from the first or best available sources. It should not be inferred that they have been disseminated only in the medium, in the language, or to the area indicated. Items from foreign language sources are translated; those from English-language sources are transcribed. Except for excluding certain diacritics, FBIS renders personal names and place-names in accordance with the romanization systems approved for U.S. Government publications by the U.S. Board of Geographic Names.

Headlines, editorial reports, and material enclosed in brackets [ ] are supplied by FBIS/JPRS. Processing indicators such as [Text] or [Excerpts] in the first line of each item indicate how the information was processed from the original. Unfamiliar names rendered phonetically are enclosed in parentheses. Words or names preceded by a question mark and enclosed in parentheses were not clear from the original source but have been supplied as appropriate to the context. Other unattributed parenthetical notes within the body of an item originate with the source. Times within items are as given by the source. Passages in boldface or italics are as published.

SUBSCRIPTION/PROCUREMENT INFORMATION

The FBIS DAILY REPORT contains current news and information and is published Monday through Friday in eight volumes: China, East Europe, Central Eurasia, East Asia, Near East & South Asia, Sub-Saharan Africa, Latin America, and West Europe. Supplements to the DAILY REPORTs may also be available periodically and will be distributed to regular DAILY REPORT subscribers. JPRS publications, which include approximately 50 regional, worldwide, and topical reports, generally contain less time-sensitive information and are published periodically.

Current DAILY REPORTs and JPRS publications are listed in *Government Reports Announcements* issued semimonthly by the National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, Virginia 22161 and the *Monthly Catalog of U.S. Government Publications* issued by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

The public may subscribe to either hardcover or microfiche versions of the DAILY REPORTs and JPRS publications through NTIS at the above address or by calling (703) 487-4630. Subscription rates will be provided by NTIS upon request. Subscriptions are available outside the United States from NTIS or appointed foreign dealers. New subscribers should expect a 30-day delay in receipt of the first issue.

U.S. Government offices may obtain subscriptions to the DAILY REPORTs or JPRS publications (hardcover or microfiche) at no charge through their sponsoring organizations. For additional information or assistance, call FBIS, (202) 338-6735,or write to P.O. Box 2604, Washington, D.C. 20013. Department of Defense consumers are required to submit requests through appropriate command validation channels to DIA, RTS-2C, Washington, D.C. 20301. (Telephone: (202) 373-3771, Autovon: 243-3771.)

Back issues or single copies of the DAILY REPORTs and JPRS publications are not available. Both the DAILY REPORTs and the JPRS publications are on file for public reference at the Library of Congress and at many Federal Depository Libraries. Reference copies may also be seen at many public and university libraries throughout the United States.